

NO-A103 201

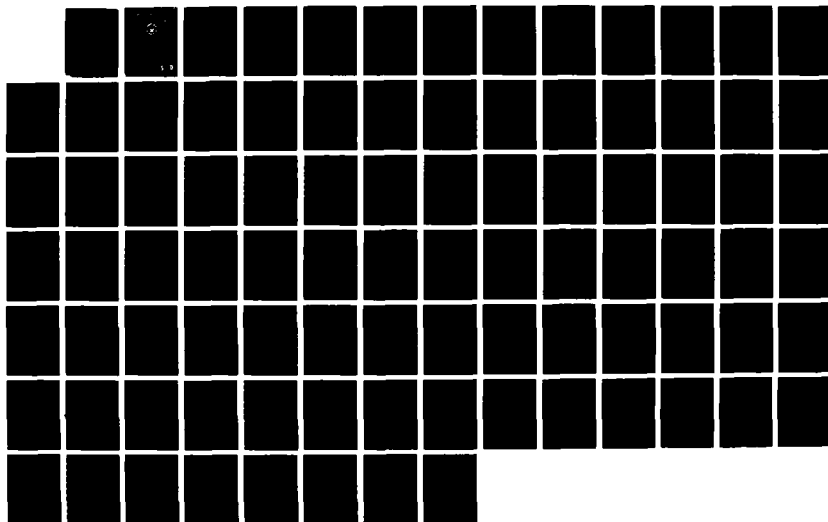
AN EXPERT SYSTEM FOR LOGISTICS FORCE DEVELOPMENT(U)  
NAVAL POSTGRADUATE SCHOOL MONTEREY CA R L CHADWICK  
JUN 87

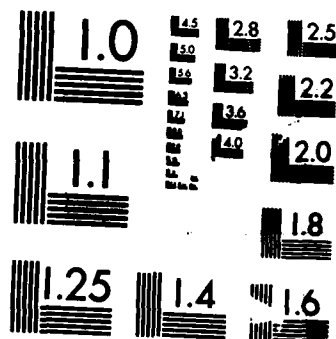
1/1

UNCLASSIFIED

F/G 15/5

NL





MICROCOPY RESOLUTION TEST CHART  
NATIONAL BUREAU OF STANDARDS-1963-A

2

# NAVAL POSTGRADUATE SCHOOL

Monterey, California

DTIC FILE COPY



AD-A183 201

## THESIS

AN EXPERT SYSTEM FOR  
LOGISTICS FORCE DEVELOPMENT

by

Robert Lawrence Chadwick

June 1987

Thesis Advisor:

Neil C. Rowe

Approved for public release; distribution is unlimited.

DTIC  
ELECTE  
AUG 17 1987  
S D  
E

87 8 14 049

unclassified

SECURITY CLASSIFICATION OF THIS PAGE

A 183201

## REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION unclassified			1b RESTRICTIVE MARKINGS		
2a SECURITY CLASSIFICATION AUTHORITY			3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.		
2b DECLASSIFICATION/DOWNGRADING SCHEDULE			5 MONITORING ORGANIZATION REPORT NUMBER(S)		
4 PERFORMING ORGANIZATION REPORT NUMBER(S)			7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School		
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School		6b OFFICE SYMBOL (If applicable) 52	7b ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000		
6c ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000		7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School			
8a NAME OF FUNDING/SPONSORING ORGANIZATION		8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER		
8c ADDRESS (City, State, and ZIP Code)		10 SOURCE OF FUNDING NUMBERS			
		PROGRAM ELEMENT NO	PROJECT NO	TASK NO	WORK UNIT ACCESSION NO
11 TITLE (Include Security Classification) AN EXPERT SYSTEM FOR LOGISTICS FORCE DEVELOPMENT					
12 PERSONAL AUTHOR(S) Chadwick, Robert Lawrence					
13a TYPE OF REPORT Master's Thesis		13b TIME COVERED FROM TO		14 DATE OF REPORT (Year Month Day) 1987 June	
15 PAGE COUNT 89					
16 SUPPLEMENTARY NOTATION Only					
17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)		
FIELD	GROUP	SUB-GROUP	logistics planning; expert systems; force development		
19 ABSTRACT (Continue on reverse if necessary and identify by block number) The purpose of this research is to show the feasibility of an expert system to aid logistics planners in determining the types and numbers of logistics units needed to support corps contingency plans. The proposed system represents the domain knowledge of logistics planners as rules and uses backwards chaining to infer the types and numbers of logistics units needed. Next a chain of command, represented as a tree diagram, is developed for the recommended units by aggregating individual units into battalions and groups. The system was implemented on a VAX 11/785 computer using the PROLOG programming language. Logistics plans for both a light corps and a heavy corps were used to test the system.					
20 DISTRIBUTION/AVAILABILITY OF ABSTRACT <input checked="" type="checkbox"/> UNCLASSIFIED/UNLIMITED <input type="checkbox"/> SAME AS RPT <input type="checkbox"/> DTIC USERS			21 ABSTRACT SECURITY CLASSIFICATION unclassified		
22a NAME OF RESPONSIBLE INDIVIDUAL Prof. Neil C. Rowe			22b TELEPHONE (Include Area Code) (408) 646-2462		22c OFFICE SYMBOL Code 52Rp

Approved for public release; distribution is unlimited.

# **An Expert System For Logistics Force Development**

by

**Robert Lawrence Chadwick**  
Captain, United States Army  
B. S., United States Military Academy, 1978

Submitted in partial fulfillment of the  
requirements for the degree of

**MASTER OF SCIENCE IN COMPUTER SCIENCE**

from the

**NAVAL POSTGRADUATE SCHOOL**

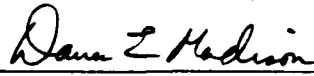
June 1987

Author:

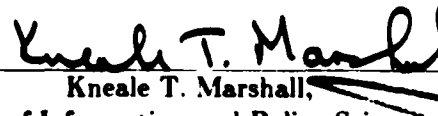
  
Robert L. Chadwick

Approved by:

  
Neil C. Rowe, Thesis Advisor

  
Dana Madison, Second Reader

  
Vincent Y. Lum, Chairman,  
Department of Computer Science

  
Kneale T. Marshall,  
Dean of Information and Policy Sciences

## ABSTRACT

The purpose of this research is to show the feasibility of an expert system to aid logistics planners in determining the types and numbers of logistics units needed to support corps contingency plans. The proposed system represents the domain knowledge of logistics planners as rules and uses backwards chaining to infer the types and numbers of logistics units needed. Next a chain of command, represented as a tree diagram, is developed for the recommended units by aggregating individual units into battalions and groups. The system was implemented on a VAX 11/785 computer using the PROLOG programming language. Logistics plans for both a light corps and a heavy corps were used to test the system.

<b>Accession For</b>	
NTIS GRA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	



## THESIS DISCLAIMER

The reader is cautioned that the programs contained in this research may not have been tested for all cases. The primary focus of the research was not to develop precise rules for determining logistics unit requirements, but rather evaluation of techniques and procedures that could be used if such information was available. While every effort was made to ensure correct programming, use of the programs without additional validations and testing is at the risk of the user.

## TABLE OF CONTENTS

	Page
I. INTRODUCTION -----	7
A. LOGISTICS PLANNING -----	7
B. EXPERT SYSTEMS -----	8
C. CURRENT EXPERT SYSTEMS APPLICATIONS -----	9
D. SCOPE -----	10
II. BACKGROUND -----	14
A. PURPOSE -----	14
B. LOGISTICS REQUIREMENTS -----	14
C. LOGISTICS UNIT SELECTION -----	16
D. CHAIN OF COMMAND -----	18
1. Mixed Strategy -----	18
2. Pure Strategy -----	19
E. BALANCED FORCE -----	20
III. IMPLEMENTATION AND DESIGN -----	23
A. SYSTEMS ORGANIZATION -----	23
B. DATA STRUCTURES -----	23
1. Lists -----	23
2. Trees -----	23



C. USER INTERFACE -----	25
1. Input -----	26
2. Output -----	27
D. KNOWLEDGE REPRESENTATION -----	28
E. INFERENCE ENGINE -----	30
F. CONFLICT RESOLUTION -----	30
G. TREE BUILDING ALGORITHM -----	32
1. Nature of Problem-----	32
2. Problem Solution -----	33
H. TREE BALANCING -----	36
IV. RESULTS -----	38
A. SYSTEM REQUIREMENTS -----	38
B. SYSTEM STATISTICS-----	38
C. TEST 1: HEAVY CORPS-----	38
D. TEST 2: LIGHT CORPS-----	39
V. CONCLUSIONS AND RECOMMENDATIONS-----	40
A. CONCLUSIONS -----	40
B. RECOMMENDATIONS-----	42
APPENDIX A: TEST 1: HEAVY CORPS -----	43
APPENDIX B: TEST 1: LIGHT CORPS -----	53
APPENDIX C: SOURCE CODE-----	60
INITIAL DISTRIBUTION LIST -----	88

## I. INTRODUCTION

### A. LOGISTICS PLANNING

The focus of this paper is the logistics planning associated with corps contingency planning for the US Army. A US Army corps is a flexible organization tailored to meet a specific mission. The corps organization normally consists of two to five combat divisions; some other combat and combat support units; and finally the logistics units needed to support the corps\*. Logistics planning for the corps begins once an estimate has been made regarding the types and numbers of combat and combat support units needed to perform the operation. With this knowledge and other mission characteristics, the logistics planner must determine the logistical requirements of the force and also the types and numbers of logistics units capable of providing the support. [Ref. 1]

Development of the logistical force to support a corps operation is a complex matter. Although there are some general guidelines on how to do this planning, there is no specific formula or algorithmic solution to the problem. Quantitative factors such as the amount of personnel and equipment, short tons of supplies to be moved and handled, and the distances between military units must all be considered. In addition to quantitative factors, more subjective factors must be

---

\*The term logistics units when used in this thesis refers to the supply, maintenance and petroleum companies normally assigned to support groups in a corps. More detailed information is contained in [Ref. 1].

considered e.g., the availability and reliability of local civilians in the area of hostilities that may be available to augment the logistic force.

Development of logistics plans is further complicated because the needed expertise is seldom concentrated in any one or even a few individuals. Even when the information is concentrated in a small group of military staff officers, the transient nature of military service makes keeping this base of knowledge intact virtually impossible.

## B. EXPERT SYSTEMS

The primary emphasis of this thesis is to explore the feasibility of using an expert systems approach to assist in the logistical planning domain. Expert systems are capable of using expert knowledge to attain high levels of performance in a narrow problem domain e.g., the determination of the logistical units to support a military operation. Expert systems also use information obtained from experts in an intelligent way to perform some task normally associated with human experts. The potential arrangements of various logistics units to support corps forces is quite large. Some method of transferring the procedures, strategies and rules used by experts into an expert system to determine logistical requirements could be valuable. [Ref 2]

### C. CURRENT EXPERT SYSTEMS APPLICATIONS

There are currently many examples of expert systems in use in both the commercial and military sectors. XCON for example, is an expert system used to configure VAX computer components into systems for Digital Equipment Corporation. It is capable of taking a customer purchase order and determining the components needed to ensure the order is filled with a system that is both consistent and complete. IMACS, a system used in a manufacturing environment, takes a customer order and produces a rough plan from which an estimate of resource requirements for the order is prepared. PROSPECTOR helps geologists evaluate the mineral potential for a region. TATR is used by the Air Force to assist planners in developing plans to attack enemy airfields. [Ref. 2]

Besides the demonstrated use of expert systems in real operating environments, there are some important characteristics of expert systems that make them attractive to the logistics planning problem. First of all, the development of the system will encapsulate a body of knowledge concerning logistics planning into an automated system. This will provide a corporate knowledge base of how experts in the field of logistics planning resolve problems. As discussed above, the transient nature of military service makes keeping a knowledgeable group of experts together for a very long time virtually impossible. The expert systems approach to the problem can ensure institutional memory is retained. Another useful feature is that expert systems can be capable of explaining their actions. Very few people are willing to blindly accept any

recommendation without some understanding of why a particular recommendation was made. Finally, an expert system can be adapted very easily to provide training to new personnel responsible for logistics planning.

#### D. SCOPE

The scope of this thesis is to determine the feasibility of using an expert system to perform logistics planning. The methodology used to determine feasibility is as follows:

- An analysis of the logistics planning problem was made to gain a better understanding of the key tasks associated with the problem domain. Discussion of how the logistics planning problem is currently performed is contained in Chapter II.
- A prototype was designed and implemented to perform logistics planning. Specifically, the prototype develops the two support groups for a corps support command. Figures 1.1 through 1.3 provide an example of a typical support command with two support groups. The support groups represent the major logistical elements of the corps and as such are a representative subset of the problem domain. Chapter III contains discussion on the design and implementation issues.
- An analysis of the prototype was performed by some limited testing. The results of the tests are discussed in Chapter IV.
- Finally, the recommendations and conclusions regarding the feasibility of using an expert system in the logistics planning environment are discussed in Chapter V.

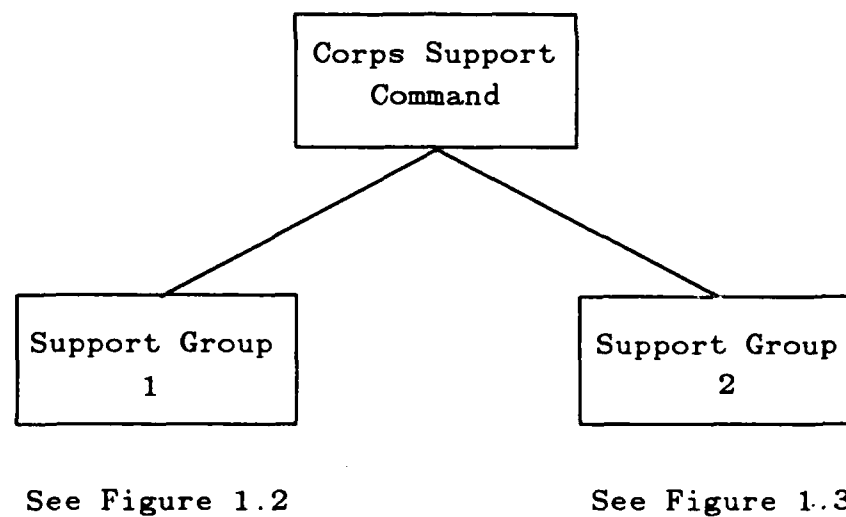
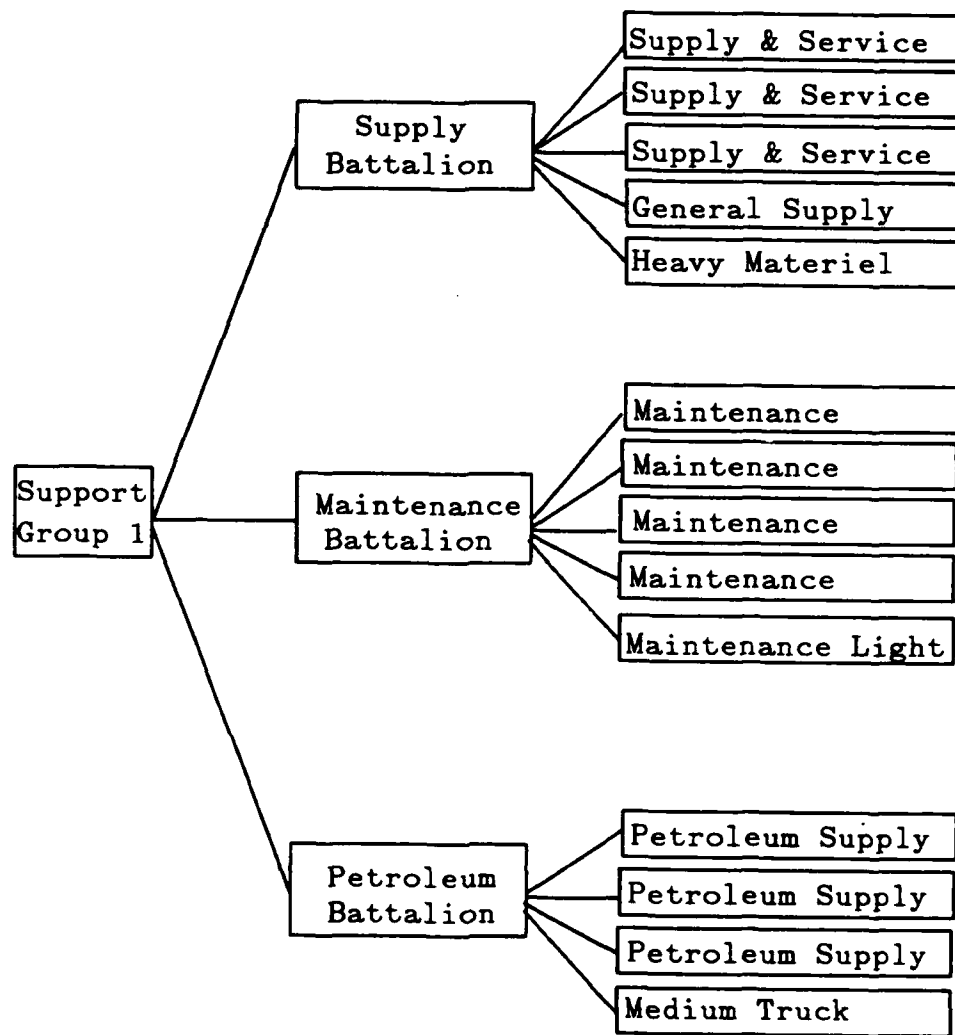
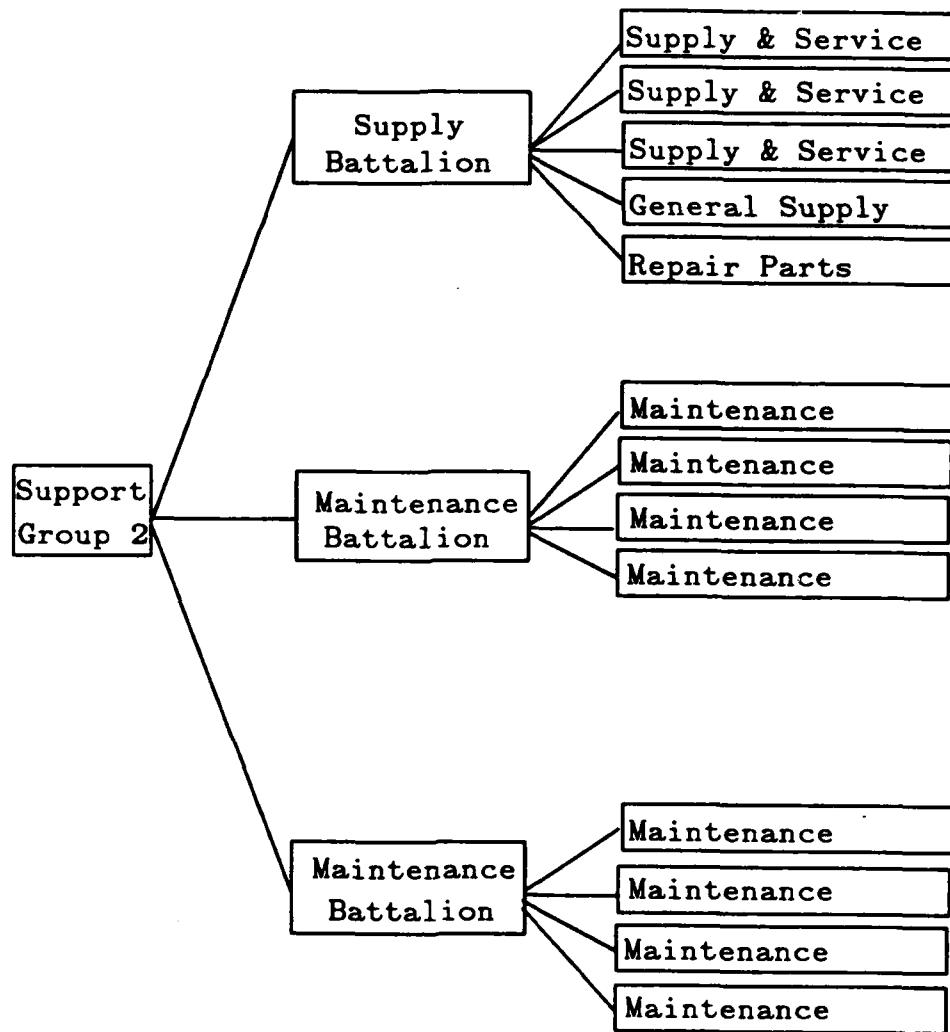


Figure 1.1 Corps Support Command



Note: All elements in this column are company-sized units.

Figure 1.2 Support Group 1



Note: All elements in this column are company-sized units.

Figure 1.3 Support Group 2



## **II. BACKGROUND**

### **A. PURPOSE**

The purpose of this chapter is to provide a better understanding of the key concepts of logistics force planning that must be understood prior to developing a prototype expert system.

### **B. LOGISTICS REQUIREMENTS**

The fundamental basis for all logistics planning is the notion of consumption factors. Consumption factors are based upon empirical data obtained from previous wars, exercises and simulations. Consumption factors express the anticipated consumption for a particular class of supply that will be consumed per soldier per day [Ref. 1]. Although there are ten categories of supplies, the most important categories are shown in Table 2.1. Using these factors, the gross daily requirements for various supply categories can be estimated by taking the product of the personnel strength of the supported force and the applicable consumption factor. The gross requirements are then normally converted to short tons.

The same methodology used to determine daily supply requirements can be extended to estimate the daily supply distribution requirements and also the storage requirements. Daily supply distribution requirements are the amount of supplies that must be distributed by logistics units to the supported force each

**TABLE 2.1 CONSUMPTION RATES**

<b>Consumption Rates (Pounds per man per day)</b>		
<b>Supply Class</b>	<b>Consumption Rate</b>	<b>Class Description</b>
I.	6.7	Subsistence Items
II.	3.3	Clothing and Individual Equipment
III.	47.8	Petroleum Products
IV.	8.8	Construction Materiel
V.	31.3	Ammunition
VI.	3.2	Personal Items
VII.	4.3	Major End Items
VIII.	0.4	Medical Items
IX.	1.52	Repair Parts

day. Daily supply distribution requirements are equivalent to the daily supply requirements since the logistics force will be expected to distribute the amount of supplies consumed each day by the force. The logistics force is also responsible for maintaining a certain level of stockage to compensate for the order and ship times associated with replenishing its supplies. The corps logistics force is authorized to maintain a certain level of stockage which is expressed in days of supply. Storage requirements are the product of daily requirements and the authorized days of supply.

It is important to note that the consumption rates in Table 2.1 are not absolute constants. Consumption rates are situationally dependent. For example, the requirement for clothing will substantially increase in a cold environment or in an environment where chemical agents are being used by the enemy forces. The logistics planner must consider these situational characteristics and make appropriate adjustments to the consumption factors.

### C. LOGISTICS UNIT SELECTION

Logistics units are designed to perform a specific functional service such as storage and issue of a particular supply category e.g., petroleum. The logistics units available to provide various services have a capability to provide a particular service per time interval. For example, a supply and service company can provide rations, clothing, construction materials and petroleum for approximately 8000 non-divisional troops per day. Some other unit capabilities are more specific, e.g., a petroleum supply company is described as having the capability to provide bulk storage for approximately one million gallons of petroleum. [Ref 1]

The logistics planner must first review the requirements discussed previously to determine the types of logistic units needed by the force. Then he must determine the numbers of each type of unit needed to provide the necessary capability to support the logistics requirements.

Unfortunately, the process of determining logistics units is not as algorithmic as the above discussion suggests. Often there is not a specific formula to determine the number of logistics units needed. Frequently, allocation rules may only state that a corps force normally has a certain number of a particular type of unit. Frequently, rules for allocating units may even conflict with one another by recommending different quantities for the same type of unit. This suggests that some form of conflict resolution must be used to select the best recommendation.

Besides general allocation guidance, the logistic planner generally has established some general heuristics concerning minimum requirements for a particular type of unit. For example, the general allocation rules may indicate a daily supply requirement for a particular supply category falls well below the daily capability of a particular unit responsible for that item. Despite the shortfall, a minimum of one unit must be present to provide the service even if the demand is quite small. Another area involves the need to have a certain amount of redundancy in the logistics structure. It may be desirable to have at least two of a particular kind of unit in the event that one unit is degraded as a result of combat conditions even though one unit may be sufficient to meet the expected workload. Finally, there may be a need to provide a service at very dispersed geographical sites. Even though one unit may be capable of providing the required requirement at a centralized location, there may be a need for the service at two sites and one unit cannot be partitioned into two effective units capable of operating at both locations.

Finally, the logistics planner may have some very precise knowledge about the number of units needed on the basis of some existing plans that have been scrutinized closely by way of simulation or perhaps have been validated on the basis of actual use. In this case actual data may indicate the need for a specific type and quantity of units and invalidate more general planning rules.

#### D. CHAIN OF COMMAND

After the numbers and types of logistics units have been determined the logistics planner is still faced with the need to provide command and control by establishing a chain of command. For example, company-sized units must first be allocated to battalions and battalions are then normally allocated to two support groups.

There are two basic strategies the logistics planner may use to develop the chain of command for the proposed force. For the purpose of discussion, assume the unit requirements listed in Table 2.2 are needed to support the logistics force. The allocation methods used to distribute these units are discussed below.

##### 1. Mixed Strategy

Logistics units are classified in two ways. The first is the general classification according to the unit's functional mission e.g., supply. The second classification is the unit's specialized support role indicating whether the unit is a general support(GS) or a direct support(DS) unit. Direct support units provide support directly to customers and are analogous to retail activities in the private sector. General support units support direct support units.

TABLE 2.2 UNIT REQUIREMENTS

Estimated Unit Requirements			
Unit Description	Functional Area	Level of Support	Quantity
Supply & Service	Supply	Direct Support	4
Field Service	Supply	General Support	1
General Supply	Supply	General Support	1

The mixed strategy provides a chain of command by organizing units into battalions on the basis of functional mission only e.g., supply. Hence battalions have a mix of direct support and general support units. This arrangement provides a more flexible battalion but it may also complicate the mission of the battalion by not focusing attention to a more specific mission. The units listed in Table 2.2 are displayed in Figure 2.1 using the mixed strategy.

## 2. Pure Strategy

The pure strategy provides a chain of command by going one step further than the mixed strategy discussed above. It organizes units on the basis of functionality as well as the units capability i.e., DS or GS. This results in battalions with a functional mission of pure direct support or general support. The units listed in Table 2.2 are displayed in Figure 2.2 using the pure strategy.

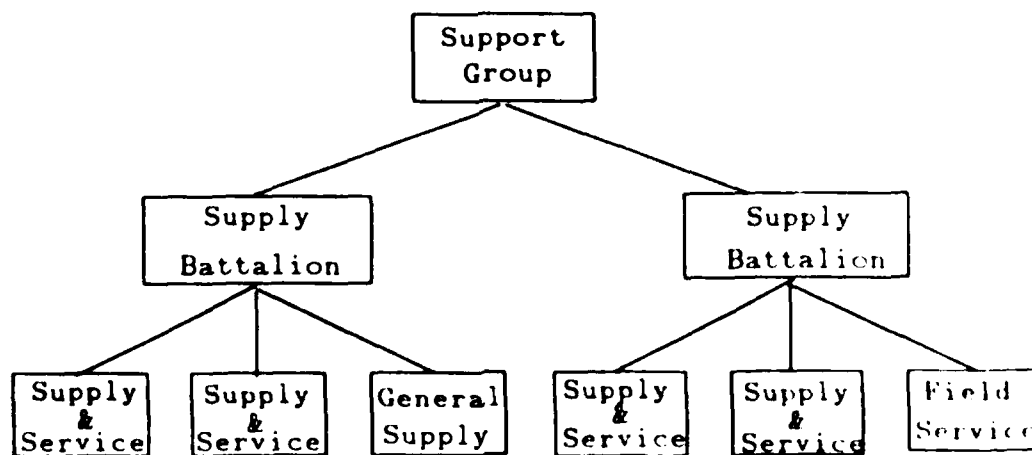


Figure 2.1 Chain of Command Using Mixed Strategy

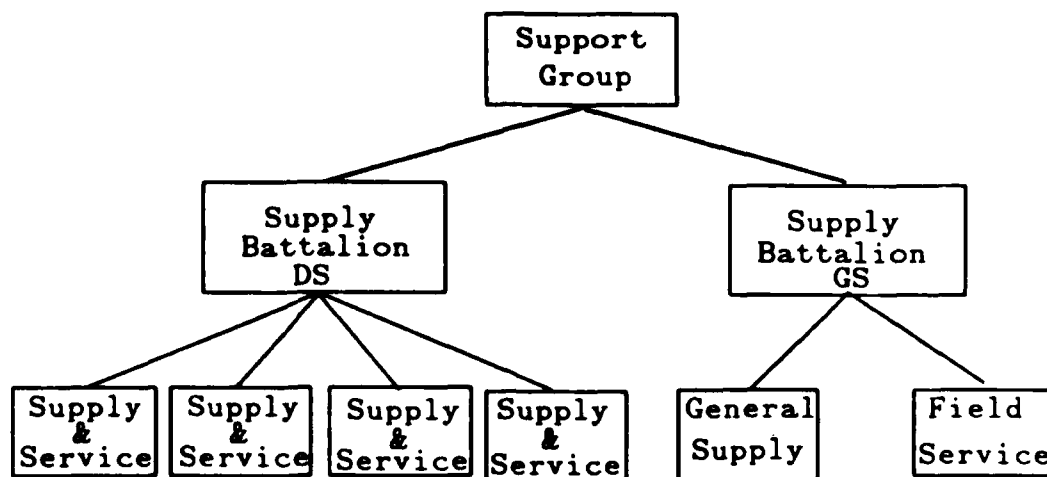


Figure 2.2 Chain of Command Using Pure Strategy

---

#### E. BALANCED FORCE

In addition to providing a chain of command, the logistics planner is concerned with structuring a logistics organization with balanced capabilities. This reflects the anticipated need to have the capability to have activities operating independently over wide areas as opposed to being all located at some central area. In general, battalions with the same mission should have the same capabilities. This is often difficult to achieve because the unit requirements estimated above may not partition into equivalent battalions. This will occur when the number of units to be partitioned is not a multiple of the number of battalions. To improve force balance, it may be desirable to adjust unit requirements in order to achieve a better partitioning. Because of the levels of

accuracy in the methods used to determine force requirements, a 20 percent adjustment factor was selected as the acceptable threshold of change for this thesis. Therefore if 15 supply and service companies were to be distributed among 4 supply battalions, the recommendation of 15 would be changed to 16 in order to allow even distribution of 4 units per battalion. It should also be noted the balancing is done by individual unit type. Two battalions must have proportionate amounts of the same type of units in order to be considered balanced, not just the same quantity of units.



### III. IMPLEMENTATION AND DESIGN

#### A. SYSTEMS ORGANIZATION

The architecture for the logistics prototype is illustrated in Figure 3.1. A brief description of each component of the architecture is described in the paragraphs below.

The collection of domain knowledge about logistics planning is contained in the knowledge base. For the prototype, an example of domain knowledge may be a particular rule used to determine if a particular logistics unit is needed to support a force.

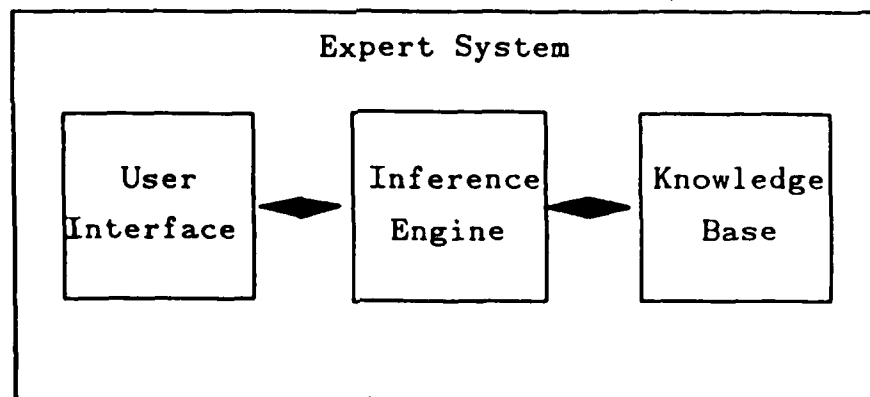


Figure 3.1 System Architecture

---

The inference engine contains an interpreter that decides how to apply the rules of the knowledge base to infer new knowledge. In addition, the inference engine contains a scheduler to determine the order in which rules should be applied. [Ref. 2]

The user interface of the prototype provides two major functions. First it provides the user a means of updating the knowledge base with force dependent facts. This allows the prototype to tailor the logistics force to a specific situation. Secondly, the user interface provides the results of using the prototype by way of an output mechanism.

## B. DATA STRUCTURES

The list and the tree were the two essential data structures needed to implement the prototype. Since the prototype was implemented in the PROLOG programming language, use of these data structures was quite convenient [Ref. 3].

### 1. Lists

A list is an ordered set of elements of finite length. An element of a list can also be a list. The first element of a list is the "head" and the remainder of the list is the "tail". The examples contained in Table 3.1 illustrate the representation of lists in PROLOG.

The list data structure is provided by the PROLOG language and the language allows the easy manipulation of lists. Any list can be represented by [X|Y] where X is the head of the list and Y is the tail of the list. By building

**TABLE 3.1 LIST REPRESENTATION**

<b>Prolog List Representation</b>		
<b>List</b>	<b>Head</b>	<b>Tail</b>
[a,b,c]	a	[b,c]
[a,[b,c],d]	a	[[b,c],d]
[a]	a	[]
[]	no head	no tail

upon this representation it is very easy to develop user defined list processing functions such as appending an item to a list, deleting an item from a list and numerous others.

## 2. Trees

A tree data structure can be easily represented by using the list data structure. The tree data structure also facilitates representation of the chain of command hierarchy of military units. In the prototype there was a need to represent a three level tree. The root of the tree is the corps support command which has two support groups as its children. The support groups have various battalions as children. Finally, the battalions will have logistics companies as children. The companies will be the leaves of the tree. The tree diagram in Figure 3.2 can be represented by the following list:

[support\_group,[supply\_bns,maintenance\_bns,petroleum\_bns]]

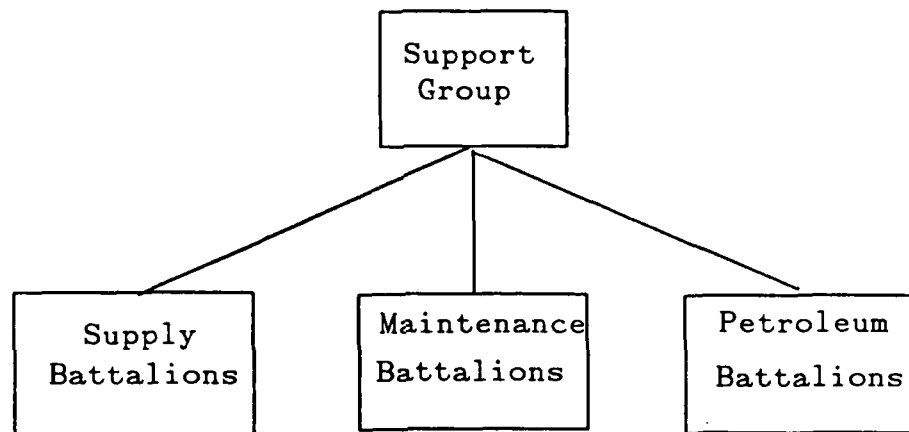


Figure 3.2 Tree Structure a Support Group

---

Besides being an excellent representation for the command and control hierarchy, previously defined list processing functions can be used to manipulate the tree. Therefore, list functions can manipulate tree data structures.

There are few comments regarding the tree data structure. The first point is that the level of nesting of lists corresponds to the level of the tree. The first level is the root of the tree and the last level of nesting represents leaves of the tree. For the purpose of the prototype, the tree is a multi-way tree, where the number of children for any tree node is at least two but less than seven [Ref.4]. This branching factor is limited because of the real-world constraint that limits a unit to commanding between two and seven units.

### C. USER INTERFACE

The user interface can be divided into the input and the output requirements of the prototype. Each of these areas is discussed below.

#### 1. Input

The prototype uses a menu-based approach to obtain user information necessary to reach conclusions regarding the requirements for logistics units. Menus provide an easy way for the user to specify the composition and the mission of the force.

The menus used in the prototype were implemented by an "ask\_which" predicate written by Professor Rowe at the Naval Postgraduate School [Ref 5]. The "ask\_which" predicate provides the user with a set of questions which can be answered by a "yes" or a "no". The user is then asked to provide the numbers to those questions which can be answered by a "yes". The "ask\_which" predicate then asserts facts into the knowledge base on the basis of user responses. Table 3.2 provides a sample menu, the user responses and the resulting facts that will be added to the knowledge base.

TABLE 3.2 USER MENU

Menu for User Input
<ol style="list-style-type: none"><li>1. Are "H" series armor divisions in the force?</li><li>2. Are "J" series armor divisions in the force?</li><li>3. Are "J" series mech divisions in the force?</li></ol> <p>Give numbers of questions whose answer is yes. [1,2].</p> <p>As a result of responding that questions 1 and 2 were true, the following facts were added to the knowledge base:</p> <p>fact(armor_h_series). fact(armor_j_series).</p>

The menu-based approach is a good method to obtain all essential facts that are relevant to any planning situation. After the initial information is obtained, the prototype will only ask specific questions that are relevant to the user's previous question responses. For example, if a user specifies that armor divisions are in the force, the system will then ask the user how many. By only asking relevant questions in view of previous responses the system is able to present a more intelligent interface to the user. The interface also uses "demons" to activate procedures to update the knowledge base. Demons are procedures that are only activated when certain conditions are satisfied. For example, when the user specifies the force will be deploying to a cold environment, the cold weather condition will be added to the knowledge base. A demon can recognize the presence of this condition and make adjustments to the logistics consumption factors that will be increased in a cold environment e.g., clothing requirements.

## 2. Output

The primary purpose of the prototype's output was to provide the user with some indication as to why specific recommendations regarding force requirements were made. For example, the prototype may recommend that as a minimum, 2 supply and service units are needed to support a corps force regardless of its size and mission. Finally, the prototype displays the chain of command for the force in a column format. All companies commanded by a particular battalion are printed in a column that is shifted to the right of the battalion. Likewise, all battalions are printed in a column to the right of their commanding support group. See Appendices A and B if additional information on output formats is needed.

## D. KNOWLEDGE REPRESENTATION

The knowledge associated with logistics planning was represented by rules. Rules were selected because of the natural way in which they represent the knowledge of logistics planners. Besides reflecting the way that many experts solve logistics problems, rules are easily implemented in the PROLOG language used for the prototype.

Rules have the format of "if condition(s) then action(s)". Table 3.3 provides some typical rules. It should be noted that some conditions can in fact be rules themselves. For example, the allocation of supply and service units is based upon the type of scenario. The scenario is determined by a scenario rule. Rules like

scenario are referred to as intermediate predicates. Intermediate predicates allow the generalization of several facts into a single concept. This abstraction greatly reduces the amount of conditions (or rule right sides) that must be listed.

The PROLOG representation of some typical rules is also provided in Table 3.3. The left sides of PROLOG rules i.e., the part before the ":-" symbol are facts if the rule right sides are true.

A final note regarding rules concerns rule partitioning. The functional nature of logistics units allows very efficient partitioning of the knowledge base. For example, rules concerning supply units can be written into a file while rules for maintenance units can be placed into another. Partitioning eases development by allowing the problem to be decomposed by the divide and conquer strategy. More importantly however, partitioning can support more efficient implementation because the inference engine only needs the rules that are relevant to the problem at hand. Thus rules concerning maintenance units need not be in main memory when the expert system is determining supply unit requirements.

#### E. INFERENCE ENGINE

A backward-chaining-control structure was chosen for the inference engine of the prototype. With backward chaining the system starts with some goal and tries to prove it by use of the rules and facts contained in the knowledge base. The selection of backward chaining was based upon two factors. First of all, the backward-chaining-control structure is provided by the PROLOG interpreter and



**TABLE 3.3 KNOWLEDGE REPRESENTATION**

<b>Knowledge Representation</b>	
<b>Rules</b>	<b>Prolog Representation</b>
If the corps is a heavy corps and the expected intensity is high then planning is for a scenario of the first type.	scenario(1) :- fact(heavy_corps), fact(high_intensity).
If the scenario is of the first type and the number of divisions is known then 3 supply and service companies are required per division.	unit(supply_and_Service_company,Required):- scenario(1), number_divisions(N), Required is N * 3.
If the scenario is of the first type the climate is not cold then 5 lbs/man per day is the planning factor for miscellaneous equipment.	factor(misc_equip,5) :- scenario(1), not(fact(cold)).

thus eliminated the development of another control structure. Secondly, backward chaining appears more efficient because it focuses on conclusions relevant to a particular planning problem. Backward chaining does not waste time searching for information unrelated to the problem.

#### **F. CONFLICT RESOLUTION**

Chapter II discussed the fact that recommendations concerning a unit may conflict. For example, one rule may recommend that only one general supply company is needed while another rule may recommend that two are needed.

Because of the potential for conflicting recommendations, the format for rules concerning unit requirements is as follows:

unit(Type\_Unit,[Number\_Recommended,Reason]) :- conditions.

The use of the above format allows the prototype to collect all recommendations regarding a particular unit as well as the reason for the recommendations. It should also be noted that when the above rule is invoked, the "Type Unit" and "Reason" are bound and "Number Recommended" is unbound until the rule succeeds. There are four possible types of recommendations: "minimum", "general", "special" and "analogy". A "minimum" type recommends a minimum number of required units regardless of mission characteristics. A "general" makes recommendations based upon a quantitative factor e.g., one unit per 8000 soldiers in the force. A "special" makes recommendations based upon the special characteristics of the mission provided by the user. Finally, an "analogy" recommends a required number by determining the number required in an existing corps force with a similar mission.

The reasons types discussed above are used to assign relative weights to the recommendations made by the prototype. Table 3.4 shows an example of some possible recommendations made by the prototype. The sum of column 4 below is divided by the sum of column 3 in order to obtain a weighted average for the recommendations i.e., 2. When recommendations are made that fall below a "minimum" recommendation, they are eliminated from the list of recommendations. The conflict-resolution strategy also performs integer division and ignores the fractional part of the division in order to determine weighted averages. A more liberal approach of performing division and rounding to the nearest whole integer value should be adopted in any future versions of the

prototype. Weighting factors are similar to global constants in traditional programming languages and may be modified quite easily.

## G. TREE BUILDING ALGORITHM

The previous sections identify how the prototype made recommendations concerning the need for particular logistics units. When conflicting recommendations are made, a conflict resolution strategy was employed to reach a final conclusion. This section concerns the arrangement of units into battalions and support groups in order to provide a chain of command to the logistics force.

### 1. Nature of Problem

As was discussed in Chapter II, a chain of command must be provided to the logistics units that ensures the optimal arrangement of units into different groups i.e., battalions and support groups. Combinatorial mathematics is the discipline normally associated with the optimal arrangement of objects [Ref. 6].

Unfortunately, traditional methods of combinatorial mathematics which guarantee optimal solutions may not be appropriate to the present problem of

TABLE 3.4 CONFLICT RESOLUTION STRATEGY

Conflict Resolution Strategy			
Recommendation	Reason	Weight Factor	Weighted Value
1	general	1	1
1	minimum	1	1
2	analogy	3	6
3	special	2	6
SUM		7	14

arranging units into battalions and support groups. This is due to the fact that enumeration of all possible solutions to the problem may require an unacceptable amount of computing time. Furthermore, it may be impossible to specify what exactly an optimum arrangement may be in quantitative terms. In view of these problems, it was determined that a heuristic approach providing an acceptable solution to the problem was needed, even though the solution may not guarantee an optimal solution.

## 2. Problem Solution

A heuristic approach to the problem was chosen that uses the concept of a "greedy" algorithm to reach a near optimal solution. Greedy algorithms ensure that choices made at each step of a solution process will not need to be retracted at a later steps of the process by backtracking. The greedy algorithm is more commonly used in problems such as transportation networks where the goal is to find the best path through a network based upon some criteria e.g., maximum or minimum capacity of the network.

The concept of the greedy algorithm was used to determine an acceptable solution to the unit arrangement problem. This was possible because the chain of command structure requires the piecemeal building of a hierarchical organizational structure or tree diagram from the bottom-up. Companies must be assigned to battalions before battalions can be assigned to support groups. At each step of this tree building process, the best assignment of units is made locally. For example, the best assignment of units is made to make balanced

battalions without regard to balancing support groups. Once battalions have been developed, the best assignment of battalions is made to balance the capabilities between two support groups.

The formal representation of the tree building algorithm is discussed in Figure 3.3. The terminology of "children" and "parent" used when discussing tree data structures is analagous to the command relationship that exists between companies(children) and battalions(parents). Likewise, battalions are children to the support groups.

The tree building algorithm is general enough that it can be used to allocate units to battalions and battalions to support groups. To better understand the allocation process, Table 3.5 shows 4 supply and service companies, 2 general supply companies and 1 repair parts company being allocated to 2 supply battalions. The algorithm also ensures that a height-balanced-multi-way tree is produced [Ref. 4].

The chain of command strategies of "mixed" and "pure" discussed in Chapter II are easily implemented by the prototype using the same tree building algorithm. The only difference between the two strategies concerns the initial step of collecting units to be distributed to parents. The "mixed" strategy collects units by functional specification only e.g., supply while the "pure" strategy collects units to be distributed by functional specification and capability e.g., supply and direct support.

- 
1. Create a list of candidate children, segregated by type, that can be assigned to the same type parents. For example, supply units can be assigned to supply battalions. All company sized units must be assigned to battalions before battalions can be assigned to support groups.
  2. Determine the number of parents needed to allocate the list of children identified in "1" above. Then create an ordered list of "N" parents that the list of children identified in "1" above can be allocated to. e.g., [[supply\_battalion1],[supply\_battalion2]...[supply\_battalionN]].
  3. Remove the first element of the children list and assign it to a parent as follows:
    - a. If all parents have an equal number of children then assign to to the first parent in the parent list.
    - b. If all parents are not equal then assign to the first parent that has one less than its immediate predecessor.
  4. Repeat step "3" until the children list is exhausted.
  5. Repeat step "1" until all children i.e., supply,maintenance and petroleum companies have been assigned parents. The last iteration should be the assignment of the battalions that have been created to two support groups.

Figure 3.3 Tree Building Algorithm

---

**TABLE 3.5 TREE BUILDING ALGORITHM EXAMPLE**

<b>Tree Building Algorithm Example</b>		
<b>Step</b>	<b>Supply Battalion 1</b>	<b>Supply Battalion 2</b>
Step 1	no units	no units
Step 2	supply & service company	
Step 3	supply & service company	supply & service company
Step 4	supply & service company supply & service company	supply & service company
Step 5	supply & service company supply & service company	supply & service company supply & service company
Step 6	supply & service company supply & service company supply & service company	supply & service company supply & service company
Step 7	supply & service company supply & service company supply & service company	supply & service company supply & service company general supply company
Step 8	supply & service company supply & service company supply & service company general supply company	supply & service company supply & service company general supply company
Step 9	supply & service company supply & service company supply & service company general supply company	supply & service company supply & service company general supply company repair parts company

**H. TREE BALANCING**

The final task of the prototype was to make adjustments to the initial recommendations so a more balanced force structure or tree could be developed. This was accomplished by adjusting initial recommendations for each type of unit so that the adjusted number represented a multiple of the number of battalions to which they would be assigned. This is done only when the adjustment is less than plus or minus 20 percent of the original recommendation provided by the prototype. For example, if 16 supply and service units are to be assigned to 3

supply battalions, the number of supply and service units would be adjusted to 15. Besides the 20 percent constraint, the adjustment was made by selecting the closest multiple regardless of whether it was an increase or decrease. Using the previous example, the numbers 18 and 12 are also multiples of three but 15 is the closest multiple.



## IV. RESULTS

### A. SYSTEM REQUIREMENTS

The prototype was implemented on a Digital Equipment Corporation VAX 11/785 minicomputer running the UNIX operating system. The prototype was written using a C-Prolog interpreter developed at the University of Edinburgh [Ref. 3].

### B. SYSTEM STATISTICS

The prototype occupies a maximum of 35.5K bytes of memory. Table 4.1 provides some run time statistics on two test cases that were using the prototype. The output listing for both test cases are contained in Appendices A and B.

### C. TEST 1: HEAVY CORPS

Appendix A contains the results of running the prototype to determine the logistical units needed to support a 6 division force consisting of heavy armor and mechanized units. As should be expected the force requires significant logistical support i.e., 54 units.

TABLE 4.1 RUN TIME RESULTS

System Run Time Results				
Test	Connect-Time Seconds	CPU-Time Seconds	Global Stack K-bytes	Local Stack K-bytes
Test 1	70	36.18	62	54
Test 2	55	20.85	25	29

#### D. TEST 2: LIGHT CORPS

Appendix B contains the results of running the prototype to determine the logistical units needed to support a 3 division force consisting of light airborne and infantry divisions. As should be expected the logistics force is considerably smaller than the previous one i.e., 27 units versus 54.

## V. CONCLUSIONS AND RECOMMENDATIONS

### A. CONCLUSIONS

The adoption of an expert systems approach to the domain of logistics planning appears to be technically feasible. In particular, an expert systems approach appears to be an excellent way to resolve logistics planning since it relies so heavily upon heuristics. The PROLOG language was an excellent tool to develop the prototype.

The translation of logistics rules or heuristics into encoded PROLOG rules was easily accomplished. Any serious implementation of an expert logistics planning system would require a substantial increase in knowledge engineering to meaningfully capture the entire complexity of the problem. Once the knowledge engineering task is adequately performed however, the task of enhancing the knowledge base can be accomplished and the resulting performance of the system improved markedly.

Although other control structures exist, the backward-chaining-control structure appears to be best suited to the task because it focuses on relevant conclusions needed to determine solutions. This could also be a benefit to future implementations since backward chaining is provided in the PROLOG language and also in most expert systems shells currently available.

The problem of arranging logistics units into a structured force i.e., battalions and groups, proved to be one of the most interesting problems of the thesis. This thesis presents one method to solve the problem. The solution technique highlights one of the most important aspects of expert systems: the ability to resolve problems by rules or heuristics when more traditional algorithmic solutions are not feasible or practical. Good heuristic solutions to allocation problems could be used in numerous application areas.

The functional nature of logistics e.g., supply, maintenance, etc. suggests that a full implementation of a logistics planning system could be developed to work in a parallel processing environment. For example, one processor could be working on supply units while another is working on maintenance units to develop the required supply and maintenance battalions needed by the force. The resulting battalions could then be allocated to the support groups by another processor. This ability could significantly increase performance of the system and reduce any criticism about the slowness of expert systems.

A full development of the system could provide extensive explanation facilities to increase the interaction between users and the system. A human expert could then override the system when recommendations and corresponding explanations do not appear to be plausible. The system could also be developed for Computer-Aided Instruction so students could gain instant feedback on the relationships between the combat force and the logistics force.

Perhaps the biggest weakness of the prototype is possible resistance of users to accept and use the system. Skeptics will be quick to point to its weaknesses and its shortfalls. This situation should not preclude adopting the system, but it does emphasize the need for good knowledge engineering to ensure that the system adequately represents the knowledge possessed by experts. It should also be recognized that the system will, like most human experts, make mistakes. Making mistakes is not a reason for abandoning the system, but an opportunity to improve the system by giving it additional knowledge to preclude repetition of the error. Despite these possible problems, an expert system(if properly implemented) can perform the tasks of experts in a rigorous way on a daily basis.

#### B. RECOMMENDATIONS

It is recommended that an enhanced prototype be developed. During development, extensive effort should be made to capture expert knowledge from domain experts. The enhanced prototype should then be tested against the experts to determine its performance and practicality.

It is also recommended that additional research be performed to determine the applicability of rule-based methods of determining solutions to allocation problems. In particular allocation or arrangement of objects into a balance tree structure should be pursued.

## APPENDIX A

### TEST 1: HEAVY CORPS

% prolog

1:Are "J" series armor divisions in the force?

2:Are "H" series armor divisions in the force?

3:Are "J" series mech divisions in the force?

4:Are "H" series mech divisions in the force?

5:Are airborne divisions in the force?

6:Are air assault divisions in the force?

7:Are motorized divisions in the force?

8:Are light infantry divisions in the force?

9:Are standard infantry divisions in the force?

Give number of only question whose answer is yes.[1,2,3].

Please enter the number of "J" series armor divisions?1.

Please enter the number of "H" series armor divisions?2.

Please enter the number of "J" series mech divisions?3.

1:Is the force to be engaged in a low intensity conflict?

2:Is the force to be engaged in a mid intensity conflict?

3:Is the force to be engaged in a hi intensity conflict?

Give number of only question whose answer is yes.[3].

1:Is the force to be depolyed to a cold environment?

2:Is the force to be deployed to a temperate climate?

3:Is the force to be deployed to a hot environment?

Give number of only question whose answer is yes.[1].

The daily force requirement for rations is: 212 short tons

The daily force requirement for misc equip is: 318 short tons

The daily force requirement for petroleum is: 2862 short tons

The daily force requirement for ammo is: 1855 short tons

The daily force requirement for barrier materiel is: 424 short tons

The daily force requirement for personnel items is: 159 short tons

The daily force requirement for major items is: 954 short tons

The daily force requirement for medical items is: 106 short tons

**REQUIREMENTS ANALYSIS FOR: heavy-materiel company**  
**INITIAL RECOMMENDATIONS:**

- [1.general]
- [1.minimum]
- [2.analogy]
- [2.special]

**REVISED LIST:**

- [1.general]
- [1.minimum]
- [2.analogy]
- [2.special]

**FINAL RECOMMENDATION:**

1

**REQUIREMENTS ANALYSIS FOR: supply-and \_service \_company**  
**INITIAL RECOMMENDATIONS:**

- [4.general]
- [6.minimum]
- [16.analogy]
- [18.special]

**REVISED LIST:**

- [6.minimum]
- [16.analogy]
- [18.special]

**FINAL RECOMMENDATION:**

15

**REQUIREMENTS ANALYSIS FOR: general-supply \_company**

**INITIAL RECOMMENDATIONS:**

- [2.general]
- [2.minimum]
- [3.general]
- [6.special]
- [8.analogy]

**REVISED LIST:**

- [2.general]
- [2.minimum]
- [3.general]
- [6.special]
- [8.analogy]

**FINAL RECOMMENDATION:**

5

**REQUIREMENTS ANALYSIS FOR: repair-parts supply company**

**INITIAL RECOMMENDATIONS:**

[1.analogy]  
[1.minimum]  
[3.general]

**REVISED LIST:**

[1.analogy]  
[1.minimum]  
[3.general]

**FINAL RECOMMENDATION:**

1

**REQUIREMENTS ANALYSIS FOR: maintenance-company\_dsfwd**

**INITIAL RECOMMENDATIONS:**

[4.minimum]  
[12.special]

**REVISED LIST:**

[4.minimum]  
[12.special]

**FINAL RECOMMENDATION:**

9

**REQUIREMENTS ANALYSIS FOR: maintenance company-dsrear**

**INITIAL RECOMMENDATIONS:**

[2.minimum]  
[6.special]

**REVISED LIST:**

[2.minimum]  
[6.special]

**FINAL RECOMMENDATION:**

4

**REQUIREMENTS ANALYSIS FOR: light-maintenance\_company\_gs**

**INITIAL RECOMMENDATIONS:**

[4.minimum]  
[12.special]

**REVISED LIST:**

[4.minimum]  
[12.special]

**FINAL RECOMMENDATION:**

9



**REQUIREMENTS ANALYSIS FOR: petroleum-supply company**

**INITIAL RECOMMENDATIONS:**

[6.minimum]

[13.general]

**REVISED LIST:**

[6.minimum]

[13.general]

**FINAL RECOMMENDATION:**

9

**REQUIREMENTS ANALYSIS FOR: medium-truck company**

**INITIAL RECOMMENDATIONS:**

[2.analogy]

[2.general]

[2.minimum]

**REVISED LIST:**

[2.analogy]

[2.general]

[2.minimum]

**FINAL RECOMMENDATION:**

2

## COMMAND AND CONTROL STRUCTURE USING MIXED ALGORITHM:

### 1st SPT GRP

#### 1st SUPPLY BN

- 1st supply-and service company
- 2nd supply-and service company
- 3rd supply-and service company
- 4th repair-parts supply company
- 5th general-supply company

#### 2nd SUPPLY BN

- 1st supply-and service company
- 2nd supply-and service company
- 3rd supply-and service company
- 4th supply-and service company
- 5th general-supply company
- 6th general-supply company

#### 1st MAINT BN

- 1st maintenance company-dsrear
- 2nd maintenance-company dsfwd
- 3rd maintenance-company dsfwd
- 4th light-maintenance company\_gs
- 5th light-maintenance company\_gs

#### 2nd MAINT BN

- 1st maintenance company-dsrear
- 2nd maintenance-company dsfwd
- 3rd maintenance-company dsfwd
- 4th light-maintenance company\_gs
- 5th light-maintenance company\_gs
- 6th light-maintenance company\_gs

#### 1st PETROL BN

- 1st petroleum-supply company
- 2nd petroleum-supply company
- 3rd petroleum-supply company
- 4th petroleum-supply company
- 5th petroleum-supply company
- 6th medium-truck company

**2nd SPT GRP**

**1st SUPPLY BN**

- 1st supply-and service company
- 2nd supply-and service company
- 3rd supply-and service company
- 4th supply-and service company
- 5th general-supply company

**2nd SUPPLY BN**

- 1st supply-and service company
- 2nd supply-and service company
- 3rd supply-and service company
- 4th supply-and service company
- 5th heavy-materiel company
- 6th general-supply company

**1st MAINT BN**

- 1st maintenance company-dsrear
- 2nd maintenance-company dsfwd
- 3rd maintenance-company dsfwd
- 4th light-maintenance company gs
- 5th light-maintenance company gs

**2nd MAINT BN**

- 1st maintenance company-dsrear
- 2nd maintenance-company dsfwd
- 3rd maintenance-company dsfwd
- 4th maintenance-company dsfwd
- 5th light-maintenance company gs
- 6th light-maintenance company gs

**1st PETROL BN**

- 1st petroleum-supply company
- 2nd petroleum-supply company
- 3rd petroleum-supply company
- 4th petroleum-supply company
- 5th medium-truck company

**THE FOLLOWING RECOMMENDATIONS WERE MADE TO BALANCE THE FORCE**

The old number for supply-and service company was changed from 15 to 16 .

The old number for general-supply company was changed from 5 to 4 .

The old number for maintenance-company dsfwd was changed from 9 to 8 .

The old number for light-maintenance company gs was changed from 9 to 8 .

The old number for petroleum-supply company was changed from 9 to 10 .

**COMMAND AND CONTROL STRUCTURE USING MIXED ALGORITHM  
AFTER BALANCING:**

**1st SPT GRP**

**1st SUPPLY BN**

- 1st supply-and service company
- 2nd supply-and service company
- 3rd supply-and service company
- 4th supply-and service company
- 5th general-supply company

**2nd SUPPLY BN**

- 1st supply-and service company
- 2nd supply-and service company
- 3rd supply-and service company
- 4th supply-and service company
- 5th heavy-materiel company
- 6th general-supply company

**1st MAINT BN**

- 1st maintenance company-dsrear
- 2nd maintenance-company dsfwd
- 3rd maintenance-company dsfwd
- 4th light-maintenance company gs
- 5th light-maintenance company gs

**2nd MAINT BN**

- 1st maintenance company-dsrear
- 2nd maintenance-company dsfwd
- 3rd maintenance-company dsfwd
- 4th light-maintenance company gs
- 5th light-maintenance company gs

**1st PETROL BN**

- 1st petroleum-supply company
- 2nd petroleum-supply company
- 3rd petroleum-supply company
- 4th petroleum-supply company
- 5th petroleum-supply company
- 6th medium-truck company

**2nd SPT GRP**

**1st SUPPLY BN**

1st supply-and service company  
2nd supply-and service company  
3rd supply-and service company  
4th supply-and service company  
5th general-supply company

**2nd SUPPLY BN**

1st supply-and service company  
2nd supply-and service company  
3rd supply-and service company  
4th supply-and service company  
5th repair-parts supply company  
6th general-supply company

**1st MAINT BN**

1st maintenance company-dsrear  
2nd maintenance-company dsfwd  
3rd maintenance-company dsfwd  
4th light-maintenance company\_gs  
5th light-maintenance company\_gs

**2nd MAINT BN**

1st maintenance company-dsrear  
2nd maintenance-company dsfwd  
3rd maintenance-company dsfwd  
4th light-maintenance company\_gs  
5th light-maintenance company\_gs

**1st PETROL BN**

1st petroleum-supply company  
2nd petroleum-supply company  
3rd petroleum-supply company  
4th petroleum-supply company  
5th petroleum-supply company  
6th medium-truck company

## COMMAND AND CONTROL STRUCTURE USING PURE ALGORITHM:

### 1st SPT GRP

#### 1st MAINT BN

- 1st light-maintenance\_company\_gs
- 2nd light-maintenance\_company\_gs
- 3rd light-maintenance\_company\_gs
- 4th light-maintenance\_company\_gs

#### 2nd MAINT BN

- 1st maintenance\_company-dsrear
- 2nd maintenance\_company-dsrear
- 3rd maintenance-company\_dsfdw
- 4th maintenance-company\_dsfdw
- 5th maintenance-company\_dsfdw
- 6th maintenance-company\_dsfdw

#### 1st SUPPLY BN

- 1st supply-and\_service\_company
- 2nd supply-and\_service\_company
- 3rd supply-and\_service\_company
- 4th supply-and\_service\_company
- 5th supply-and\_service\_company

#### 2nd SUPPLY BN

- 1st repair-parts\_supply\_company
- 2nd heavy-materiel\_company
- 3rd general-supply\_company
- 4th general-supply\_company
- 5th general-supply\_company
- 6th general-supply\_company

#### 1st PETROL BN

- 1st petroleum-supply\_company
- 2nd petroleum-supply\_company
- 3rd petroleum-supply\_company
- 4th petroleum-supply\_company
- 5th petroleum-supply\_company
- 6th medium-truck\_company

2nd SPT GRP

1st MAINT BN

1st light-maintenance\_company\_gs  
2nd light-maintenance\_company\_gs  
3rd light-maintenance\_company\_gs  
4th light-maintenance\_company\_gs

2nd MAINT BN

1st maintenance\_company-dsrear  
2nd maintenance\_company-dsrear  
3rd maintenance-company\_dsfdw  
4th maintenance-company\_dsfdw  
5th maintenance-company\_dsfdw  
6th maintenance-company\_dsfdw

1st SUPPLY BN

1st supply-and\_service\_company  
2nd supply-and\_service\_company  
3rd supply-and\_service\_company  
4th supply-and\_service\_company  
5th supply-and\_service\_company

2nd SUPPLY BN

1st supply-and\_service\_company  
2nd supply-and\_service\_company  
3rd supply-and\_service\_company  
4th supply-and\_service\_company  
5th supply-and\_service\_company  
6th supply-and\_service\_company

1st PETROL BN

1st petroleum-supply\_company  
2nd petroleum-supply\_company  
3rd petroleum-supply\_company  
4th petroleum-supply\_company  
5th petroleum-supply\_company  
6th medium-truck\_company

[ Prolog execution halted ]

## APPENDIX B

### TEST 2: LIGHT CORPS

% prolog

1:Are "J" series armor divisions in the force?

2:Are "H" series armor divisions in the force?

3:Are "J" series mech divisions in the force?

4:Are "H" series mech divisions in the force?

5:Are airborne divisions in the force?

6:Are air assault divisions in the force?

7:Are motorized divisions in the force?

8:Are light infantry divisions in the force?

9:Are standard infantry divisions in the force?

Give number of only question whose answer is yes.[5.9].

Please enter the number of airborne divisions?1.

Please enter the number of standard infantry divisions?2.

1:Is the force to be engaged in a low intensity conflict?

2:Is the force to be engaged in a mid intensity conflict?

3:Is the force to be engaged in a hi intensity conflict?

Give number of only question whose answer is yes.[3].

^M

1:Is the force to be depoloyed to a cold environment?

2:Is the force to be deployed to a temperate climate?

3:Is the force to be deployed to a hot environment?

Give number of only question whose answer is yes.[2].

^M

The daily force requirement for rations is: 98 short tons

The daily force requirement for misc equip is: 122 short tons

The daily force requirement for petroleum is: 1151 short tons

The daily force requirement for ammo is: 612 short tons

The daily force requirement for barrier materiel is: 196 short tons

The daily force requirement for personnel items is: 73 short tons

The daily force requirement for major items is: 367 short tons

The daily force requirement for medical items is: 24 short tons

^M



REQUIREMENTS ANALYSIS FOR: heavy-materiel\_company  
INITIAL RECOMMENDATIONS:

- [1.analogy]
- [1.general]
- [1.minimum]
- [1.special]

REVISED LIST:

- [1.analogy]
- [1.general]
- [1.minimum]
- [1.special]

FINAL RECOMMENDATION:

1

REQUIREMENTS ANALYSIS FOR: supply-and\_service\_company  
INITIAL RECOMMENDATIONS:

- [1.minimum]
- [2.general]
- [6.special]
- [8.analogy]

REVISED LIST:

- [1.minimum]
- [2.general]
- [6.special]
- [8.analogy]

FINAL RECOMMENDATION:

5

REQUIREMENTS ANALYSIS FOR: general-supply\_company  
INITIAL RECOMMENDATIONS:

- [1.general]
- [1.minimum]
- [1.special]
- [4.analogy]

REVISED LIST:

- [1.general]
- [1.minimum]
- [1.special]
- [4.analogy]

FINAL RECOMMENDATION:

2

REQUIREMENTS ANALYSIS FOR: repair-parts\_supply\_company  
INITIAL RECOMMENDATIONS:

[1.analogy]

[1.minimum]

[3.general]

REVISED LIST:

[1.analogy]

[1.minimum]

[3.general]

FINAL RECOMMENDATION:

1

REQUIREMENTS ANALYSIS FOR: maintenance-company \_dsfwd

INITIAL RECOMMENDATIONS:

[3.special]

[4.minimum]

[10.analogy]

REVISED LIST:

[4.minimum]

[10.analogy]

FINAL RECOMMENDATION:

8

REQUIREMENTS ANALYSIS FOR: maintenance \_company-dsrear

INITIAL RECOMMENDATIONS:

[1.special]

[2.minimum]

[5.analogy]

REVISED LIST:

[2.minimum]

[5.analogy]

FINAL RECOMMENDATION:

4

REQUIREMENTS ANALYSIS FOR: light-maintenance \_company \_gs

INITIAL RECOMMENDATIONS:

[2.minimum]

[3.special]

REVISED LIST:

[2.minimum]

[3.special]

FINAL RECOMMENDATION:

2

REQUIREMENTS ANALYSIS FOR: petroleum-supply \_company

INITIAL RECOMMENDATIONS:

[1.minimum]

[5.general]

REVISED LIST:

[1.minimum]

[5.general]

FINAL RECOMMENDATION:

3

REQUIREMENTS ANALYSIS FOR: medium-truck \_company

INITIAL RECOMMENDATIONS:

[1.general]

[1.minimum]

[2.analogy]

REVISED LIST:

[1.general]

[1.minimum]

[2.analogy]

FINAL RECOMMENDATION:

1

## COMMAND AND CONTROL STRUCTURE USING MIXED ALGORITHM:

### 1st S I T GRP

#### 1st SUPPLY BN

- 1st supply-and \_service \_company
- 2nd supply-and \_service \_company
- 3rd supply-and \_service \_company
- 4th heavy-materiel \_company
- 5th general-supply \_company

#### 1st MAINT BN

- 1st maintenance \_company-dsrear
- 2nd maintenance \_company-dsrear
- 3rd maintenance-company \_dsfwd
- 4th maintenance-company \_dsfwd
- 5th light-maintenance \_company \_gs

#### 1st PETROL BN

- 1st petroleum-supply \_company
- 2nd petroleum-supply \_company
- 3rd petroleum-supply \_company
- 4th medium-truck \_company

### 2nd SPT GRP

#### 1st SUPPLY BN

- 1st supply-and \_service \_company
- 2nd supply-and \_service \_company
- 3rd repair-parts \_supply \_company
- 4th general-supply \_company

#### 1st MAINT BN

- 1st maintenance \_company-dsrear
- 2nd maintenance-company \_dsfwd
- 3rd maintenance-company \_dsfwd
- 4th maintenance-company \_dsfwd

#### 2nd MAINT BN

- 1st maintenance \_company-dsrear
- 2nd maintenance-company \_dsfwd
- 3rd maintenance-company \_dsfwd
- 4th maintenance-company \_dsfwd
- 5th light-maintenance \_company \_gs

**THE FOLLOWING RECOMMENDATIONS WERE MADE TO BALANCE THE FORCE:**

The old number for supply-and service company was changed from 5 to 6 .

The old number for maintenance-company\_dsfwd was changed from 8 to 9 .

The old number for maintenance\_company-dsrear was changed from 4 to 3 .

**COMMAND AND CONTROL STRUCTURE USING MIXED ALGORITHM**

**AFTER BALANCING:**

**1st SPT\_GRP**

**1st SUPPLY BN**

1st supply-and service company

2nd supply-and service company

3rd supply-and service company

4th heavy-materiel company

5th general-supply company

**1st MAINT BN**

1st maintenance\_company-dsrear

2nd maintenance-company\_dsfwd

3rd maintenance-company\_dsfwd

4th maintenance-company\_dsfwd

5th light-maintenance\_company\_gs

**1st PETROL BN**

1st petroleum-supply company

2nd petroleum-supply company

3rd petroleum-supply company

4th medium-truck company

**2nd SPT\_GRP**

**1st SUPPLY BN**

1st supply-and service company

2nd supply-and service company

3rd supply-and service company

4th repair-parts supply company

5th general-supply company

**1st MAINT BN**

1st maintenance\_company-dsrear

2nd maintenance-company\_dsfwd

3rd maintenance-company\_dsfwd

4th maintenance-company\_dsfwd

**2nd MAINT BN**

1st maintenance\_company-dsrear

2nd maintenance-company\_dsfwd

3rd maintenance-company\_dsfwd

4th maintenance-company\_dsfwd

5th light-maintenance company\_gs

## COMMAND AND CONTROL STRUCTURE USING PURE ALGORITHM:

### 1st SPT GRP

#### 1st MAINT BN

- 1st maintenance\_company-dsrear
- 2nd maintenance\_company-dsrear
- 3rd maintenance-company dsfwd
- 4th maintenance-company dsfwd
- 5th maintenance-company dsfwd
- 6th maintenance-company dsfwd

#### 1st SUPPLY BN

- 1st supply-and service\_company
- 2nd supply-and service\_company
- 3rd supply-and service\_company
- 4th supply-and service\_company
- 5th supply-and service\_company
- 6th supply-and service\_company

#### 1st PETROL BN

- 1st petroleum-supply\_company
- 2nd petroleum-supply\_company
- 3rd petroleum-supply\_company
- 4th medium-truck\_company

### 2nd SPT GRP

#### 1st MAINT BN

- 1st light-maintenance\_company\_gs
- 2nd light-maintenance\_company\_gs

#### 2nd MAINT BN

- 1st maintenance\_company-dsrear
- 2nd maintenance-company dsfwd
- 3rd maintenance-company dsfwd
- 4th maintenance-company dsfwd
- 5th maintenance-company dsfwd
- 6th maintenance-company dsfwd

#### 1st SUPPLY BN

- 1st repair-parts\_supply\_company
- 2nd heavy-materiel\_company
- 3rd general-supply\_company
- 4th general-supply\_company

[ Prolog execution halted ]

## APPENDIX C

### SOURCE CODE

/\*The "logistic\_force" predicate is the top level predicate used by the logistics planning system. The first call is to load the system via the "load\_system" predicate. After all necessary files have been loaded the user is queried for the composition of force via the "force\_list" predicate. Next the user is asked for information about unique mission characteristics via the "environmental\_factors" predicate. Finally, after all necessary input data has been provided to the system, a series of calls "requirements" is made to determine the actual type and quantity of logistic units that are needed. Command and control of selected units is determined by the "tree" predicate.\*/

logistic\_force :-

```
    prog([load_system,
    force_list,
    environmental_factors,
    doall(calculate),
    requirements(heavy-materiel_company,supply.gs),
    requirements(supply-and_service_company,supply.ds),
    requirements(general-supply_company,supply.gs),
    requirements(repair-parts_supply_company,supply.gs),
    requirements(maintenance-company_dsfdw,maintenance.ds),
    requirements(maintenance_company-dsrear,maintenance.ds),
    requirements(light-maintenance_company_gs.maintenance.gs),
    requirements(petroleum-supply_company,petroleum.gs),
    requirements(medium-truck_company,petroleum.gs),
    clean_dbase,tree(mixed,1),balance_tree,tree(mixed,2),
    tree(pure,3)]).
```

/\* This procedure loads all necessary files for the logistics planning system \*/

load\_system :-

```
    consult(estimator), consult(scenario),
    consult(factors), consult(demons),
    consult(ask), consult(ask1),
    consult(ask2), consult(explain),
    consult(questions), consult(materiel),
    consult(supply), consult(repair),
    consult(supply2), consult(ds_maint_rear).
```

```

consult(ds_maint_fwd), consult(gs_maint_light).
consult(petrol), consult(balance).
consult(tree), consult(lists).

/*The following predicates are used to query the user by means of a menu to
determine the force list and the mission characteristics of the corps force. When
triggered the "demon" predicate causes the further questioning of the user based
upon initial responses to the first menu. */
force_list :-
    ask_which([armor_j_series,armor_h_series,mech_j_series,
    mech_h_series,airborne,air_assault,motorized,light_infantry,
    infantry]),doall(demon),nl,nl.

environmental_factors :-
    ask_which([low,mid,hi]), nl,nl,
    ask_which([cold,temperate,hot]),nl,nl.

/*This predicate readjusts initial recommendations so that a more balanced force
can be assembled.*/
balance_tree:-
    balance(heavy-materiel_company.supply,gs).
    balance(supply-and_service_company.supply,ds).
    balance(general-supply_company.supply,gs).
    balance(repair-parts_supply_company.supply,gs).
    balance(maintenance-company_ds_fwd,maintenance.ds).
    balance(maintenance_company_ds_rear,maintenance.ds).
    balance(light-maintenance_company_gs,maintenance,gs).
    balance(petroleum-supply_company.petroleum,gs).
    balance(medium-truck_company.petroleum,gs) ,nl.

/* "prog" is equivalent of a Lisp PROG construct: it does a list of predicate
expressions in order, ignoring whether they succeed or fail, without permitting any
backtracking between them. */
prog([]).
prog([P|L]) :- not(negcall(P)), prog(L).
negcall(P) :- not(poscall(P)).
poscall(P) :- call(P).
poscall(P).
/*"doall" is used to ensure all possible calls to a predicate have i.e., it forces
backtracking. */
doall(P) :- not(alltried(P)).
alltried(P) :- call(P),fail.

```



```

/*This file is used to estimate the actual unit requirements of the Corps Support
Command. The "requirements" predicate calls upon the rule base by means of
the "setof" predicate. Because of conflicting recommendations, conflict resolution
is provided by the "weighted_avg" predicate. This predicate weights conclusions
based upon whether they were based on specific, general, or analagous
information about the proposed force that is being planned.*/

```

```

/*This predicate collects the set of all recommendations concerning a type of unit.
The set is a lists of list: each containing the number of units that were
recommended and the basis for the recommendation.*/

```

```

requirements(Type_of_Unit,Category,Level) :-
    setof(X,unit(Type_of_Unit,X),L),
    write('REQUIREMENTS ANALYSIS FOR: '), write(Type_of_Unit).nl.nl,
    write('INITIAL RECOMMENDATIONS:'),nl,writelist(L).nl,
    write('REVISED LIST:'),nl, check(L,L1), writelist(L1).nl,
    weighted_avg(L1,Type_of_Unit,Category,Level,Weighted_Avg).nl,
    write('FINAL RECOMMENDATION: '),nl,
    write('    ').write(Weighted_Avg).nl.nl.

```

```

/*This predicate deletes all recommendations failing to meet minimum reqmts. */
check(Initial_List,New_List) :-
    member([Value,minimum],Initial_List),
    delete2(Value,Initial_List,New_List).

```

```

/*This predicate determines the weighted average of all unit recommendations*/
weighted_avg(L,Type_of_Unit,Category,Level,Weighted_Avg) :-
    weighted_sum(L,Weighted_Sum,Divisor),
    Weighted_Avg is ( Weighted_Sum // Divisor),
    build_troop_list(Weighted_Avg,Type_of_Unit,Category,Level).

```

```

/*This predicate asserts a fact "troop list" for each recommended unit.*/
build_troop_list(0,AnyType,AnyCategory,Level).
build_troop_list(Number,Type_of_Unit,Category,Level) :-
    N is Number -1,
    asserta(troop_list(Type_of_Unit,Category,Level)),
    build_troop_list(N,Type_of_Unit,Category,Level).

```

/\* This predicate is used by the "weighted\_avg" predicate above. It sums the list of weighted unit recommendations. \*/

weighted\_sum([],0.0).

weighted\_sum([First|Rest],Total,Divisor):-

weighted\_unit\_sum(First,Number,Number1).

weighted\_sum(Rest,Totall,Divisor1).

Divisor is Number1 + Divisor1, Total is Number + Totall.

weighted\_unit\_sum([First,general],First,1).

weighted\_unit\_sum([First,minimum],First,1).

weighted\_unit\_sum([First,analogy],Number,3) :- Number is (First \* 3).

weighted\_unit\_sum([First,special],Number,2) :- Number is (First \* 2).

/\*The "tree" predicate is used to build the command and control structure of the support command. It takes all company sized units and forms battalion. Then it takes all battalion sized elements and forms the support groups.\*/

/\*The first "tree" predicate structures the force into battalions with the same functional mission e.g.. supply. It does matter what level of support the unit provides. Direct and general support units may be in the same battalion.\*/

```
tree(mixed,H) :-
    bagof(X,Level^troop_list(X.supply,Level),L).keysort(L.LS1),
    bagof(Y,Level^troop_list(Y.maintenance,Level),L2).keysort(L2.LS2),
    bagof(Z,Level^troop_list(Z.petroleum,Level),L3).keysort(L3.LS3),
    mixed(supply.LS1,L1_out),
    mixed(maintenance.LS2,L2_out),
    mixed(petroleum.LS3,L3_out),
    append(L2_out,L1_out,L4),
    append(L3_out,L4,L5),
    mixed(spt_grp.L5,L5_out),
    heading(H,Heading). write(Heading),nl,nl.
    pp(L5_out.5),update2.
```

/\*This "tree" predicate structures the force into battalions with the same functional mission and also separates direct support units from general support units.\*/

```
tree(pure,H) :-
    bagof(X,troop_list(X.supply,gs).L1).keysort(L1.L1S),
    bagof(X,troop_list(X.supply,ds).L2).keysort(L2.L2S),
    bagof(Y,troop_list(Y.maintenance,ds).L3).keysort(L3.L3S),
    bagof(Y,troop_list(Y.maintenance,gs).L4).keysort(L4.L4S),
    bagof(Z,troop_list(Z.petroleum,gs).LP).keysort(LP.LPS),
    pure(supply,gs.L1S,L1_out),
    pure(supply,ds.L2S,L2_out),
    pure(maintenance,ds.L3S,L3_out),
    pure(maintenance,gs.L4S,L4_out),
    pure(petroleum,gs.LPS,LPS_out),
    append(L1_out,L2_out,L5),
    append(L3_out,L4_out,L6),
    append(L5,L6,L7),
    append(LPS_out,L7,L7_out),
    pure(spt_grp.dsgs.L7_out,Lfinal),
    heading(H,Heading). write(Heading),nl,nl.
    pp(Lfinal.5),update2.
```

/\*The "pure" predicate forms direct or general support battalions to serve as parent units for direct or general support companies respectively.\*/

pure(Type.Level.L1.New\_List) :-

span\_of\_control(Type.Number),  
number\_pure\_battalions(Type.Level.Number.Number1),  
create\_battalions(Number1.Type.Bn\_List),  
distribute(L1.Bn\_List,New\_List).

/\*The "mixed" predicate forms battalions on the basis of functional missions such as supply. Both direct and general support units with the same functional mission can be assigned to the same battalion.\*/

mixed(Type.L1.New\_List) :-

span\_of\_control(Type.Number),  
number\_battalions(Type.Number.Number1),  
create\_battalions(Number1.Type.Bn\_List),  
distribute(L1.Bn\_List,New\_List).

/\*The following predicate determines the number of battalions needed to command and control a specified number of companies.\*/

number\_battalions(spt\_grp.Number,2).

number\_battalions(Type.Number.Number1) :-

not(command(Type.AnyNumber)),  
bagof(X.Level^troop\_list(X.Type.Level),L),  
length(L.Len),  
factor(Len.Number.Number1),  
asserta(command(Type.Number1)).

number\_battalions(Type.Number.Number1) :-

command(Type.Number1).

number\_pure\_battalions(spt\_grp.dsgs,Number,2).

number\_pure\_battalions(Type.Level.Number.Number1) :-

bagof(X.troop\_list(X.Type.Level),L),  
length(L.Len),  
factor(Len.Number,Number1).

/\*The "distribute" predicate assigns a list of candidate units to a parent headquarters. For example maintenance companies are distributed among maintenance battalions.\*/

distribute([],Bn\_List.Bn\_List) :-!

distribute([X|Tail].Bn\_List.Bn\_List2) :-

add(X.Bn\_List.Bn\_List1),  
distribute(Tail.Bn\_List1.Bn\_List2).

```

add(X.Bn_List,Bn_List_New) :-
    equal(Bn_List),
    add_first(X,Bn_List,Bn_List_New).

```

```

add(X.Bn_List,New_Bn_List) :-
    not(equal(Bn_List)),
    first_small(Bn_List,[H,L]),append([X].L,N),
    replace([H,L],[H,N].Bn_List.New_Bn_List).

```

```

add_first(X,[[Head.Tail|| Bn_List],Bn_List_New) :-
    append([X].Tail,New),
    append([[Head,New]],Bn_List,Bn_List_New).

```

```

first_small([X.Y|L].X) :-
    second(X.T1).second(Y.T2),
    length(T1.N).length(T2.N1), N1 > N.
first_small([X.Y|L].Y) :-
    second(X.T1).second(Y.T2),
    length(T1.N).length(T2.N1), N > N1.
first_small([X.Y|L].Out) :-
    second(X.T1).second(Y.T2),
    length(T1.N).length(T2.N1), N1 = N,
    first_small([Y|L].Out).

```

```

/* This predicate creates battalions in order to distribute company company sized
elements to the resulting battalions.*/

```

```

create_battalions(0,_1,[]).
create_battalions(Number,Type,[[Type.[Number]]| Bn_List]) :-
    N is Number - 1,
    create_battalions(N,Type,Bn_List).

```

```

/* These are facts that determine number of units per type battalion.*/ nf
span_of_control(supply,5). span_of_control(maintenance,5).
span_of_control(spt_grp,2). span_of_control(petroleum,5).

```

```

/* These are facts that determine which headings to use as output to inform the
user of which algorithm was used to determine command and control */
heading(1.'COMMAND AND CONTROL STRUCTURE USING MIXED ALGORITHM:').
heading(2.'COMMAND AND CONTROL STRUCTURE USING MIXED ALGORITHM AFTE
heading(3.'COMMAND AND CONTROL STRUCTURE USING PURE ALGORITHM:').

```

```
/*The "balance" predicate attempts to adjust the recommendation for a specific
type of unit by increasing or decreasing the initial recommendation to the nearest
multiple of the number of battalions. If the adjustment is within an acceptable
tolerance of 20 percent, the adjustment is made so that battalions with the same
mission have equal capabilities.*/
```

```
balance(Type_of_Unit,Category,Level) :-
    bagof(Type_of_Unit,Level^troop_list(Type_of_Unit,Category,Level).List),
    length(List,Number_Units).command(Category,Number_Bns).
    closest_multiple(Number_Bns,Number_Units,Ans),
    check_threshold(Number_Units,Ans,Ans1),
    adjust(Ans1.Type_of_Unit,Category,Level),
    revisions(Ans1,Number_Units,Type_of_Unit).
```

```
check_threshold(Number_Units,Ans,Ans) :-
    Ans > 0.
    Max_Change is (Number_Units * 0.25),
    Ans =< Max_Change.
```

```
check_threshold(Number_Units,Ans,Ans) :-
    Ans < 0.
    NewAns is (Ans * -1),
    Max_Change is (Number_Units * 0.25),
    NewAns =< Max_Change.
```

```
check_threshold(Number_Units,Ans,0).
```

```
revisions(0,Number_Units,Type_of_Unit).
revisions(Ans,Number_Units,Type_of_Unit) :-
    New_Number_Units is (Ans + Number_Units),
    write('The old number for ').write(Type_of_Unit).
    write(' was changed from ').write(Number_Units).
    write(' to ').write(New_Number_Units).write(' ').nl.
```

```

closest_multiple(Number_Bns,Number_Units,Ans) :-
    Number_Units =< Number_Bns, Ans is (Number_Bns - Number_Units).
closest_multiple(Number_Bns,Number_Units,0) :-
    R is (Number_Units mod Number_Bns). R = 0.
closest_multiple(Number_Bns,Number_Units,Ans) :-
    Lb is ((Number_Units // Number_Bns) * Number_Bns),
    Ub is (((Number_Units + Number_Bns) // Number_Bns) * Number_Bns).
    D1 is Number_Units - Lb. D2 is Ub - Number_Units,
    min2(D1,D2,Ans).

min2(D1,D2,-D1):- D1 < D2 .
min2(D1,D2,D2) :- D1 >= D2.

adjust(0,Type,Category,Level) :- !.
adjust(Adj_Qty,Type,Category,Level) :-
    Adj_Qty < 0, V1 is Adj_Qty + 1,
    retract(troop_list(Type,Category,Level)).
    adjust(V1,Type,Category,Level).

adjust(Adj_Qty,Type,Category,Level) :-
    Adj_Qty > 0. V1 is Adj_Qty - 1,
    asserta(troop_list(Type,Category,Level)).
    adjust(V1,Type,Category,Level).

```

/\*The "pp" predicate prints the hierarchical diagram for the support group. It accepts the representation of the support group as a list of lists. It indents for each level of the tree.\*/

```
pp([H|T],I) :- !, J is I + 5, pp(H,J), ppx(T,J), nl.
```

```
pp(X,I) :-
    not(member(X,[maintenance,supply,petroleum,spt_grp])),
    not(number(X)), tab(I).
    retract(fb(unit,L,Name)), write(L),write('-'), write(X),
    NL is L + 1, asserta(fb(unit,NL,Name)),nl.
```

```
pp(spt_grp,I) :-
    fb(spt_grp,2,'SPT_GRP'), tab(I), write(2), write('-'),
    write('SPT_GRP'), update2,nl.
```

```
pp(X,I) :-
    tab(I), retract(fb(X,L,Name)),
    write(L), write('-'), write(Name).
    NL is L + 1, asserta(fb(X,NL,Name)),nl.
```

```
pp(X,I) :- tab(I), number(X), nl.
```

```
ppx([],_) :- retract(fb(unit,L,' ')), asserta(fb(unit,1,' ')).
ppx([H|T],I) :- pp(H,I), ppx(T,I).
```

```
update2 :-
    retract(fb(maintenance,L1,'MAINT BN')),
    asserta(fb(maintenance,1,'MAINT BN')),
    retract(fb(supply,L2,'SUPPLY BN')),
    asserta(fb(supply,1,'SUPPLY BN')),
    retract(fb(petroleum,L3,'PETROL BN')),
    asserta(fb(petroleum,1,'PETROL BN')),
    retract(fb(spt_grp,L4,'SPT_GRP')),
    asserta(fb(spt_grp,1,'SPT_GRP')),
    retract(fb(unit,L5,' ')), asserta(fb(unit,1,' ')).
```

```
fb(unit,1,' ').
fb(maintenance,1,'MAINT BN').
fb(supply,1,'SUPPLY BN').
fb(petroleum,1,'PETROL BN').
fb(spt_grp,1,'SPT_GRP').
```



/\*The following rules are for selection of Supply and Service Companies.\*/

unit(supply-and\_service\_company,[Number\_of\_Units,special]) :-  
    scenario(1).  
    number\_divisions(N).  
    Number\_of\_Units is  $N * 3$ .

unit(supply-and\_service\_company,[Number\_of\_Units,special]) :-  
    scenario(2).  
    number\_divisions(N).  
    Number\_of\_Units is  $N * 2$ .

unit(supply-and\_service\_company,[Number\_of\_Units,special]) :-  
    scenario(3).  
    number\_divisions(N).  
    Number\_of\_Units is  $N * 1$ .

unit(supply-and\_service\_company,[Number\_of\_Units,special]) :-  
    scenario(4).  
    number\_divisions(N).  
    Number\_of\_Units is  $N * 1$ .

unit(supply-and\_service\_company,[Number\_of\_Units,minimum]) :-  
    heavy\_corps.  
    number\_divisions(N).  
    Number\_of\_Units is  $N * 1$ .

unit(supply-and\_service\_company,[Number\_of\_Units,minimum]) :-  
    light\_corps.  
    number\_divisions(N).  
    Number\_of\_Units is  $(N // 2)$ .

unit(supply-and\_service\_company,[Number\_of\_Units,analogy]):-  
    number\_divisions(N).  
     $N > 1, N < 5$ ,  
    Number\_of\_Units is 8.

unit(supply-and\_service\_company,[Number\_of\_Units,analogy]):-  
    number\_divisions(N).  
     $N = 4, N < 8$ ,  
    Number\_of\_Units is 16.

```

unit(supply-and_service_company,[Number_of_Units,general]) :-
    corps_strength(X).
    Strength is (X // 3).
    factor(Strength.8,Number_of_Units).!.

unit(supply-and_service_battalion,[Number_of_Units,general]) :-
    Number_of_Units is 2.

/*The following rules are for selection of General Supply Companies.*/
unit(general-supply_company,[Number_of_Units,special]) :-
    scenario(1).
    number_divisions(N),
    Number_of_Units is (N * 1).

unit(general-supply_company,[Number_of_Units,special]) :-
    scenario(2).
    number_divisions(N).
    factor(N.2.Number_of_Units).

unit(general-supply_company,[Number_of_Units,general]) :-
    number_divisions(N).
    factor(N.3.Number_of_Units).

unit(general-supply_company,[Number_of_Units,analogy]) :-
    number_divisions(N).
    N > 2. N < 5. Number_of_Units is 4.

unit(general-supply_company,[Number_of_Units,analogy]) :-
    number_divisions(N).
    N > 4. N < 8. Number_of_Units is 8.

unit(general-supply_company,[2,minimum]) :- heavy_corps.

unit(general-supply_company,[1,minimum]) :- light_corps.

unit(general-supply_company,[Number_of_Units,general]) :-
    corps_strength(Total). factor(rations,N1).
    factor(misc_equip.N2). factor(barrier_materiel.N3).
    factor(personnel_items.N4). R is (N1 + N2 + N3 + N4).
    General_Supply_Reqmnt is ((R * Total) // 2).
    factor(General_Supply_Reqmnt.350.Number_of_Units).!.

```

/\* The following rules are for selection of Heavy Materiel Companies. \*/

unit(heavy-materiel\_company,[2,special]) :-  
    scenario(1) .

unit(heavy-materiel\_company,[1,special]) :-  
    scenario(2) .

unit(heavy-materiel\_company,[1,special]) :-  
    scenario(3).

unit(heavy-materiel\_company,[1,special]) :-  
    scenario(4) .

unit(heavy-materiel\_company,[1,analogy]) :-  
    light\_corps .

unit(heavy-materiel\_company,[2,analogy]) :-  
    heavy\_corps .

unit(heavy-materiel\_company,[1,minimum]).

unit(heavy-materiel\_company,[Number\_of\_Units,general]) :-  
    corps\_strength(Number).  
    factor(Number,85.Number\_of\_Units)!..

/\* The following rules are for selection of Repair Parts Companies. \*/

unit(repair-parts\_supply\_company,[1,minimum]).

unit(repair-parts\_supply\_company,[1,analogy]).

unit(repair-parts\_supply\_company,[Number\_of\_Units,general]) :-  
    fact(no\_gs\_repair).factor(10.3.Number\_of\_Units).

unit(aircraft\_missile\_repair-parts\_company,1).

/\* The following rules are for selection of Maintenance Companies. \*/

unit(maintenance\_company-dsrear,[5.analogy]) :-  
    number\_divisions(N).  
     $N \geq 2, N \leq 5$ .

unit(maintenance\_company-dsrear,[Number of Units,special]) :-  
    scenario(1).  
    number\_divisions(N).  
    Number of Units is  $N + 1$ .

unit(maintenance\_company-dsrear,[Number of Units,special]) :-  
    scenario(2).  
    number\_divisions(N).  
    factor( $N \cdot 2$ ,Number of Units).

unit(maintenance\_company-dsrear,[Number of Units,special]) :-  
    scenario(3).  
    number\_divisions(N).  
    factor( $N \cdot 2$ ,Number of Units).

unit(maintenance\_company-dsrear,[Number of Units,special]) :-  
    scenario(4).  
    number\_divisions(N).  
    factor( $N \cdot 2$ ,Number of Units).

unit(maintenance\_company-dsrear,[2.minimum]).

unit(maintenance-company\_dsfd,[10.analogy]) :-  
    number\_divisions(N).  
     $N \geq 2, N \leq 5$ .

unit(maintenance-company\_dsfd,[Number of Units,special]) :-  
    scenario(1).  
    number\_divisions(N).  
    Number of Units is  $N + 2$ .

unit(maintenance-company\_dsfd,[Number of Units,special]) :-  
    scenario(2).  
    number\_divisions(N).  
    Number of Units is  $N$ .

unit(maintenance-company\_dsfd,[Number of Units,special]) :-  
    scenario(3).

```

    number_divisions(N).
    Number_of_Units is N.

unit(maintenance-company_dsfd,[Number_of_Units,special]) :-
    scenario(4),
    number_divisions(N),
    Number_of_Units is N.

unit(maintenance-company_dsfd,[4,minimum]).

unit(light-maintenance_company_gs,[10,analogy]) :-
    number_divisions(N),
    heavy_corps.
    N > 2, N < 5 .

unit(light-maintenance_company_gs,[5,analogy]) :-
    number_divisions(N),
    light-corps.
    N > 2, N < 5 .

unit(light-maintenance_company_gs,[Number_of_Units,special]) :-
    scenario(1),
    number_divisions(N),
    Number_of_Units is N + 2.

unit(light-maintenance_company_gs,[Number_of_Units,special]) :-
    scenario(2),
    number_divisions(N),
    Number_of_Units is N .

unit(light-maintenance_company_gs,[Number_of_Units,special]) :-
    scenario(3),
    number_divisions(N),
    Number_of_Units is N.

unit(light-maintenance_company_gs,[Number_of_Units,special]) :-
    scenario(4),
    number_divisions(N),
    Number_of_Units is N.

unit(light-maintenance_company_gs,[2,minimum]):- light_corps.

unit(light-maintenance_company_gs,[4,minimum]):- heavy_corps.

```

/\* The following rules are for selection of Petroleum Companies \*/  
unit(petroleum-supply\_company.[Number of Units.minimum]) :-  
heavy\_corps.number\_divisions(Number of Units).

unit(petroleum-supply\_company.[Number of Units.minimum]) :-  
light\_corps.number\_divisions(N).  
Number of Units is (N // 2).

unit(petroleum-supply\_company.[Number of Units.general]) :-  
corps\_strength(X).  
factor(petroleum.Y).  
days\_of\_supply(Z).  
Total Gallons is ((X \* Y \* Z \* 300) // 2).  
factor(Total Gallons.1000000.Number of Units),!.

days of supply(15).

unit(medium-truck\_company.[2.analogy]).  
unit(medium-truck\_company.[Number of Units.minimum]) :-  
bagof(X.troop\_list(petroleum-supply\_company.petroleum.gs),L).  
length(L,Len). factor(Len.5.Number of Units).

unit(medium-truck\_company.[Number of Units.general]) :-  
corps\_strength(X).  
factor(petroleum.Y).  
Total Daily Gallons is ((X \* Y \* 300) // 2).  
factor(Total Daily Gallons.450000.Number of Units),!.

/'The following rules are used to determine the applicable scenario' /

scenario(1) :-  
    fact(hi),  
    heavy corps.!

scenario(2) :-  
    fact(hi),  
    light corps.!

scenario(3) :-  
    fact(mid),  
    heavy corps.!

scenario(4) :-  
    fact(mid),  
    light corps.!

scenario(5) :-  
    fact(lo),  
    heavy corps.!

scenario(6) :-  
    fact(lo),  
    light corps.!

heavy corps :-  
    bagof(X.heavy(X).L).sum(L,S).  
    bagof(Y.light(Y).L1).sum(L1,S1).  
    S = S1.

light corps :-  
    bagof(X.heavy(X).L).sum(L,S).  
    bagof(Y.light(Y).L1).sum(L1,S1).  
    S = S1.

heavy corps :-  
    heavy(X).not(light(Y)).

light corps :-  
    light(X).not(heavy(X)).

```
/* This file provides explanations to assist users in answering questions */  
/* that they may not fully understand */
```

```
explain(heavy_corps) :-
```

```
    write('A heavy corps force is one that consists ').nl.  
    write('of armor and mechanized infantry divisions.').nl.  
    write('The force may have other types of divisions').nl.  
    write('but the majority of its divisions should be').nl.  
    write('armor and mechanized infantry.').
```

```
explain(light_corps) :-
```

```
    write('A light corps force is one that consists of').nl.  
    write('of light, airborne, or airmobile divisions.').nl.  
    write('The force may have other types of divisions').nl.  
    write('but the majority of its divisions should be').nl.  
    write('light, airorne, or airmobile divisions.').
```

```
explain(developed_country) :-
```

```
    write('A developed nation is a nation that has an').nl.  
    write('industrialized economy. The term implies').nl.  
    write('the nation has a developed infrastructure').nl.  
    write('of roads, rail, airports, etc..').
```

```
explain(underdeveloped_country) :-
```

```
    write('An underdeveloped nation is one in which').nl.  
    write('industrialized characteristics such as').nl.  
    write('good road and rail facilities and aiport').nl.  
    write('are not present.').
```

```
explain(independent_force) :-
```

```
    write('An independent force is one in which the').nl.  
    write('corps is expected to deploy and conduct').nl.  
    write('conduct combat operations without any').nl.  
    write('accompanying friendly forces.').
```

```
explain(commz) :-
```

```
    write('A communications zone (commz) exists ').nl.  
    write('or will exist if a theater army and its').nl.  
    write('associated support units will be present').nl.  
    write('in the area of operations providing support').nl.  
    write('to corps support activities.').
```



explain(strength) :-

write('The personnel strength estimate should be the').nl.  
write('cumulative strength for all divisional and all'). nl.  
write('nondivisional forces in the proposed corps force.').

explain(number of divisions) :-

write('The number of divisions should only include the').nl.  
write('actual number of divisions or division equivalents').nl.  
write('contained in the actual corps force.').

explain(non divisional strength) :-

write('Nondivisional strength should only include the ').nl.  
write('personnel strengths of the nondivisiona units in the corps.').

```
/* This file contains a series of ask predicates that will only be */  
/* triggered based upon a certain force composition provided by the */  
/* the user during a previous menu session */
```

```
demon :- fact(armor j series).  
    write('Please enter the number of "J" series armor divisions?').  
    read(Ans).  
    Strength is 17 * Ans.  
    asserta(strength(Strength)).  
    asserta(heavy(Ans)).
```

```
demon :- fact(armor h series).  
    write('Please enter the number of "H" series armor divisions?').  
    read(Ans).  
    Strength is 19 * Ans.  
    asserta(strength(Strength)).  
    asserta(heavy(Ans)).
```

```
demon :- fact(mech j series).  
    write('Please enter the number of "J" series mech divisions?').  
    read(Ans).  
    Strength is 17 * Ans.  
    asserta(strength(Strength)).  
    asserta(heavy(Ans)).
```

```
demon :- fact(mech h series).  
    write('Please enter the number of "H" series mech divisions?').  
    read(Ans).  
    Strength is 19 * Ans.  
    asserta(strength(Strength)).  
    asserta(heavy(Ans)).
```

```
demon :- fact(airborne).  
    write('Please enter the number of airborne divisions?').  
    read(Ans).  
    Strength is 17 * Ans.  
    asserta(strength(Strength)).  
    asserta(light(Ans)).
```

```
demon :- fact(motorized). write('Please enter the number of motorized divisions?').  
    read(Ans). Strength is 14 * Ans.  
    asserta(strength(Strength)).  
    asserta(light(Ans)).
```

```

demon :- fact(air_assault).
        write('Please enter the number of motorized divisions?').
        read(Ans). Strength is 19 * Ans.
        asserta(strength(Strength)).
        asserta(light(Ans)).

demon :- fact(light_infantry).
        write('Please enter the number of light infantry divisions?').
        read(Ans). Strength is 10 * Ans.
        asserta(strength(Strength)).
        asserta(light(Ans)).

demon :- fact(infantry).
        write('Please enter the number of standard infantry divisions?').
        read(Ans).
        Strength is 16 * Ans.
        asserta(strength(Strength)).
        asserta(light(Ans)).

demon:-
        bagof(X.heavy(X).L).sum(L,S).
        bagof(X.light(X).L1).sum(L1,S1).
        S3 is (S + S1).
        asserta(number_divisions(S3)).

demon:-
        bagof(X.heavy(X).L).sum(L,S).
        asserta(number_divisions(S)).

demon:-
        bagof(X.light(X).L).sum(L,S).
        asserta(number_divisions(S)).

demon:-
        bagof(Strength.strength(Strength).L).
        sum(L,Corps_strength). asserta(corps_strength(Corps_strength)).

sum([],0).
sum([X|L],Total) :- sum(L,Total2).Total is Total2 + X.

```

```
/*This file contains planning factors which are adjusted based upon */  
/*the operational characteristics of the particular scenario*/
```

```
factor(rations.4).
```

```
factor(misc_equip.5) :- scenario(1).not(fact(cold)).  
factor(misc_equip.5) :- scenario(2).not(fact(cold)).  
factor(misc_equip.4) :- scenario(3).  
factor(misc_equip.4) :- scenario(4).  
factor(misc_equip.4) :- scenario(5).  
factor(misc_equip.4) :- scenario(6).  
factor(misc_equip.6) :- scenario(1). fact(cold).  
factor(misc_equip.6) :- scenario(2). fact(cold).  
factor(misc_equip.4) :- not(scenario(X)).
```

```
factor(petroleum.54) :- scenario(1).  
factor(petroleum.47) :- scenario(2).  
factor(petroleum.47) :- scenario(3).  
factor(petroleum.40) :- scenario(4).  
factor(petroleum.40) :- scenario(5).  
factor(petroleum.34) :- scenario(6).  
factor(petroleum.47) :- not(scenario(X)).
```

```
factor(ammo.35) :- scenario(1).  
factor(ammo.25) :- scenario(2).  
factor(ammo.30) :- scenario(3).  
factor(ammo.25) :- scenario(4).  
factor(ammo.25) :- scenario(5).  
factor(ammo.20) :- scenario(6).  
factor(ammo.30) :- not(scenario(X)).
```

```
factor(barrier_materiel.8).
```

```
factor(personnel_items.3).
```

```
factor(major_items.18) :- scenario(1).  
factor(major_items.15) :- scenario(2).  
factor(major_items.15) :- scenario(3).  
factor(major_items.12) :- scenario(4).  
factor(major_items.12) :- scenario(5).  
factor(major_items.9) :- scenario(6).  
factor(major_items.15) :- not(scenario(X)).
```

```
factor(medical_items,1) :- not(scenario(1)).  
factor(medical_items,2) :- scenario(1).
```

```
/*The following procedures compute the gross tons of supplies needed*/  
/*per day in the corps area.*/
```

```
calculate :-
```

```
    corps_strength(N),  
    factor(X,N1),  
    Requirement is ((N * N1)//2),  
    write('The daily force requirement for '),  
    write(X).write(' is: ').  
    write(Requirement). write(' short tons').nl.
```

```

/*The following predicates are used to query the user by means of a menu to
determine the force list and the mission characteristics of the corps force.
When triggered the "demon" predicate causes the further questioning of the
user based upon initial responses to the first menu. */

```

```

force_list :-

```

```

    ask_which([armor_j_series.armor_h_series,mech_j_series.
    mech_h_series.airborne,air_assault.motorized,light_infantry,
    infantry]),doall(demon).

```

```

environmental_factors :-

```

```

    nl.nl.
    ask_which([low.mid,hi]),
    nl.nl.
    ask_which([cold.temperate.hot]),
    nl.nl.

```

```

/*This file contains question codes used by the ask predicate. In particular*/
/*this file has questioncodes applicable to logistics force planning. These*/
/*questioncodes allow terminal prompts to obtain specific information con- */
/*cerning the characteristics of a particular corps force so that logistical */
/*unit requirements can be determined*/

```

```

/*The following questioncodes help determine the appropriate scenario*/

```

```

questioncode(commz,'Is a communications zone established in the area').
questioncode(developed_country,'Is the area of operations in a developed country').
questioncode(independent_corps,'Is the force an independent corps').
questioncode(cold,'Is the force to be depoloyed to a cold environment').
questioncode(temperate,'Is the force to be deployed to a temperate climate').
questioncode(hot,'Is the force to be deployed to a hot environment').
questioncode(low,'Is the force to be engaged in a low intensity conflict').
questioncode(mid,'Is the force to be engaged in a mid intensity conflict').
questioncode(hi,'Is the force to be engaged in a hi intensity conflict').

```

```

/*The following question codes assist in determining force composition*/
questioncode(armor_h_series,'Are "H" series armor divisions in the force').
questioncode(armor_j_series,'Are "J" series armor divisions in the force').
questioncode(mech_h_series,'Are "H" series mech divisions in the force').
questioncode(mech_j_series,'Are "J" series mech divisions in the force').

```

questioncode(infantry,'Are standard infantry divisions in the force').  
questioncode(airborne,'Are airborne divisions in the force').  
questioncode(air\_assault,'Are air assault divisions in the force').  
questioncode(motorized,'Are motorized divisions in the force').  
questioncode(light\_infantry,'Are light infantry divisions in the force').

```

/* The following rules assist question-asking. The "ask" predicate */
/* asks the user a question, providing extra explanation if the user */
/* types a question mark, and returns the answer (after caching it). */
/* The "questioncode" and "explain" predicates must be provided by */
/* the programmer: the first decodes questions, and the second provides */
/* additional information for particular questions when the user has */
/* trouble understanding. The "askif" predicate handles yes/no questions: */
/* it succeeds if the user answers positively, fails if the user answers */
/* negatively, and prompts if the user answers anything else. */
/* Warning: do "abolish(asked.2)" to erase memory before a new problem. */

```

```

askif(Qcode) :- ask(Qcode.A), positive_answer(Qcode,A).
askifnot(Qcode) :- not(askif(Qcode)).
positive_answer(Qcode,A) :- affirmative(A).
positive_answer(Qcode,A) :- not(negative(A)), not(affirmative(A)).
    write('Please answer yes or no.'). read(A2), retract(asked(Qcode.A)).
    asserta(asked(Qcode.A2)), affirmative(A2).
ask(Qcode.A) :- asked(Qcode.A).
ask(Qcode.A) :- not(asked(Qcode.A)), questioncode(Qcode.Q), write(Q).
    write('? '). read(A2), ask2(Q,Qcode.A2,A).
ask2(Q,Qcode,'?'.A) :- explain(Qcode).nl.ask(Qcode,A).
ask2(Q,Qcode,A,A) :- not(A='?'), asserta(asked(Qcode.A)).
affirmative(yes).
affirmative(y).
affirmative(ye).
affirmative(right).
affirmative(ok).
affirmative(uhhuh).
negative(no).
negative(n).
negative(not).
negative(never).
negative(impossible).
negative(haha).

```



```
ask_which([A.B.C.D.E.F.G.H.I|L]) :-
    screen_ask_which([A.B.C.D.E.F.G.H.I],[A.B.C.D.E.F.G.H.I]),
    ask_which(L).
```

```
ask_which([]).
```

```
ask_which(L) :-
    length(L,N), N < 10, N > 0,
    screen_ask_which(L,L).
```

```
screen_ask_which([X|L],L2) :-
    length(L,N),
    length(L2,N2),
    N3 is N2 - N,
    write(N3),
    write(':').
    questioncode(X,Q),
    write(Q),
    write('?').
    nl.
    asserta(asked(X.no)),
    screen_ask_which(L,L2).
```

```
screen_ask_which([],L2) :-
    write('Give number of only question whose answer is yes.').
    read(AL),
    create_facts(AL,L2),
    nl.
```

```
create_facts([N|L],L2) :-
    item(N,L2,I),
    assertz(fact(I)),
    retract(asked(I.no)),
    asserta(asked(I.yes)),
    create_facts(L,L2).
create_facts([],L2) :-
    not(item(N,L2,I)),
    create(facts(L,L2)).
create_facts([],L2).
```

```
item(1,[X|L],X).
item(N,[X|L],I) :-
    N > 1, N2 is N - 1, item(N2,L,I).
```

## LIST OF REFERENCES

1. Combined Arms and Services Staff School. *Basic Logistical Principals* . Fort Leavenworth, KS, 1983.
2. Waterson, D.A.. *A Guide to Expert Systems* . Addison-Wesley, 1986.
3. Clocksin, W.F. and Mellish, C.S., *Programming in Prolog* . 2nd ed., Springer-Verlag, 1984.
4. Reingold, E.M. and Hansen, W.J., *Data Structures* . Little, Brown and Company, 1983.
5. Rowe, N.C.. *AI Through Prolog* . Prentice-Hall, 1987.
6. Foulds, L.R.. *Combinatorial Optimization for Undergraduates* . Springer-Verlag, 1984.
7. Grimaldi, R.P., *Discrete and Combinatorial Mathematics* . Addison-Wesley, 1985.

## INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2. Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3. Professor Neil C. Rowe, Code 52RP Department of Computer Science Naval Postgraduate School Monterey, California 93943-5000	2
4. Major Dana Madison 52 Department of Computer Science Naval Postgraduate School Monterey, California 93943-5000	2
5. Captain Robert Chadwick 520 Fernwood Drive Toronto, Ohio 43964	2

END

9-87

Dtic